1985-03

A database system for monitoring labor costs in a Public Works environment

Dinwiddie, David Paul

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A DATABASE SYSTEM FOR MONITORING LABOR COSTS IN A PUBLIC WORKS ENVIRONMENT

By

David Paul Dinwiddie

September 1987

Thesis Advisor: Norman R. Lyons

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) Code 54	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) A DATABASE SYSTEM FOR MONITORING LABOR COSTS IN A PUBLIC WORKS ENVIRONMENT (u)	
12 PERSONAL AUTHOR(S) Dinwiddie, David Paul	
13a TYPE OF REPORT Master's Thesis 13b TIME COVERED FROM _____ TO _____ 14 DATE OF REPORT (Year, Month, Day) 1987 September 15 PAGE COUNT 120	
16 SUPPLEMENTARY NOTATION	
17 COSATI CODES FIELD / GROUP / SUB-GROUP	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Database; dBASE; Public works; Labor; Civil Engineer; Reimbursables; Decision Support; Cost allocation; Computer; CEC

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The Naval Postgraduate School Public Works Department must monitor labor hours charged by employees to assist in managing the payroll and executing the budget. Entering data into a local system, reconciling locally kept records with official records, and transferring data to official systems is too expensive. This thesis describes the design, analysis and implementation of a prototype system capable of providing year-to-date labor information with sufficient detail and accuracy to support the Public Works Officer in his budget execution endeavors. It also addresses the problem of integrating the prototype with larger systems which currently require manual input of the same data.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Norman R. Lyons	22b TELEPHONE (Include Area Code) (408) 646-2666 22c OFFICE SYMBOL Code 54Lb

Approved for public release; distribution is unlimited.

A Database System for Monitoring Labor Costs in a Public Works Environment

by

David Paul Dinwiddie
Lieutenant, Civil Engineer Corps, United States Navy
B.S., Rose-Hulman Institute of Technology, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

ABSTRACT

The Naval Postgraduate School Public Works Department must monitor labor hours charged by employees to assist in managing the payroll and executing the budget. Entering data into a local system, reconciling locally kept records with official records, and transferring data to official systems is too expensive. This thesis describes the design, analysis and implementation of a prototype system capable of providing year-to-date labor information with sufficient detail and accuracy to support the Public Works Officer in his budget execution endeavors. It also addresses the problem of integrating the prototype with larger systems which currently require manual input of the same data.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author gratefully acknowledges the Triune God, Father, Son and Holy Spirit who made possible my engagement in this endeavor, the family of believers at Covenant Presbyterian Church whose prayers supported this effort and encouraged me to keep reasonable priorities and my lovely wife Susan who played the role of "computer widow" during the development of the prototype.

There were many people who helped in the development of this project. Those deserving specific mention include Public Works Officer LCDR Richard E. Burgoyne, Public Works Data Processing Manager John T. Perry, Administrative Officer Jeane Benton and Data Entry Clerk Ms. Cora Patricio who did most of the testing of the prototype.

A great deal of background information was provided by the NAVFAC BEST Project Manager Mr. Don A. Allen, Mr. Richard W. McDermed of CESO, and the Naval Postgraduate School Deputy Comptroller Mr. Robert Jay.

I. INTRODUCTION

A. BACKGROUND

1. Funding Policy

Current government funding policy which places increasing fiscal responsibilities at lower levels of management has brought not only increased flexibility to middle managers, but also a requirement for more sophisticated methods of decision support. To understand more fully the implications of this flexibility, it is helpful to review briefly the means by which local activities are funded.

Congress through the means of public laws (Appropriation Acts) assigns funds to agencies for specific, previously authorized programs. After these funds or appropriations are released to agencies, the Office of Management and Budget (OMB) apportions the funds to DoD and limits the obligations which may be incurred during the fiscal year. Funds are then allocated from DoD to the Comptroller of the Navy, from the Comptroller of the Navy to CNO and from CNO to major claimants for distribution to responsibility centers such as Naval Postgraduate School. Responsibility centers are authorized to incur obligations within a specified amount. [Ref. 1:pp. I-4 to I-11]

Operations and Maintenance, Navy (O&M,N) funds are subdivided by the responsibility centers and given as operating targets (OPTARS) to their cost centers. A cost center is a subdivision of a responsibility center the responsibility for which is generally assigned to one Supervisor. A local management code (LMC) is a subdivision of a cost center broken down by purpose or organization. At Naval Postgraduate School, a significant source of funds flow indirectly to the command through reimbursable jobs. A reimbursable is a lateral flow of resources from other

Government activities to finance services provided by a host in compliance with a host-tenant agreement between the activities. The host identifies the source of funds used to accomplish reimbursable work for the tenant with a four character segment number (SEG).

2. Mission Areas

a. Naval Postgraduate School

The mission of the U.S. Naval Postgraduate School is stated as follows:

> To conduct and direct the advanced education of commissioned officers, and to provide such other technical and professional instruction as may be prescribed to meet the needs of the Naval Service; and in support of the foregoing, to foster and encourage a program of research in order to sustain academic excellence. [Ref. 2:p. 6]

b. The Public Works Officer

In support of the school's mission, the Public Works Officer (PWO) is responsible to the Assistant Director of Military Operations (Fig. 1.1) for providing NPS as well as various tenant commands with maintenance, utilities and transportation support.

Partially because of provisions of the anti-deficiency act prohibiting the over-expenditure of funds, the PWO has staffed his organization (Fig. 1.2) with an Administrative Officer (AO) who is tasked with administration, coordination and direction of Public Works budget, finance and organizational methods and procedures. The position description of the Administration Officer identifies the AO as being responsible for budget formulation and presentation and for advising on the status and availability of funds as well as the capability of the department to meet objectives with available resources.

Resources available for support provided to Naval Postgraduate School are constrained by the annual operating targets. Resources available for those reimbursable services provided to tenant commands are constrained by the amount of

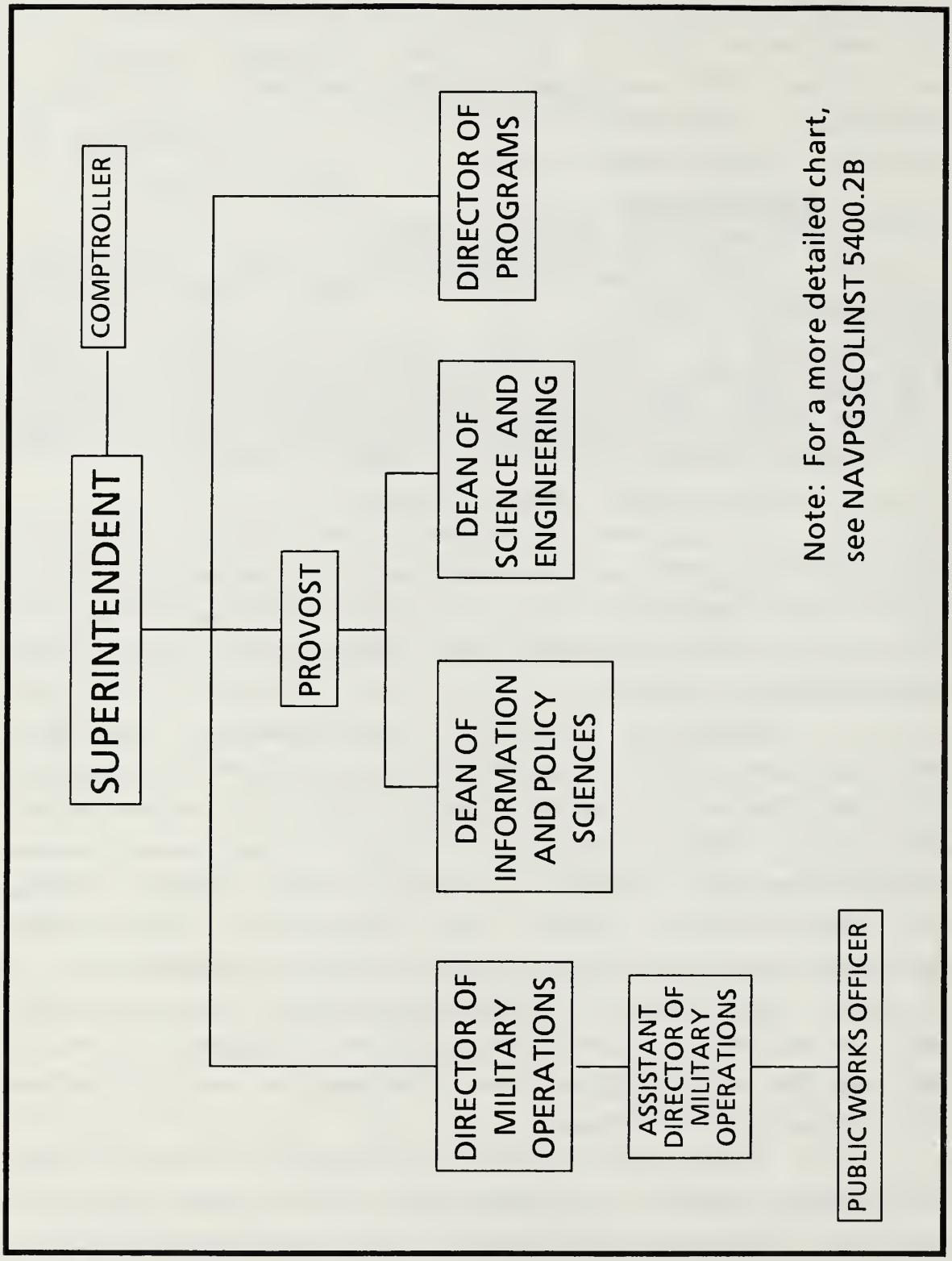


Figure 1.1 Naval Postgraduate School Organization

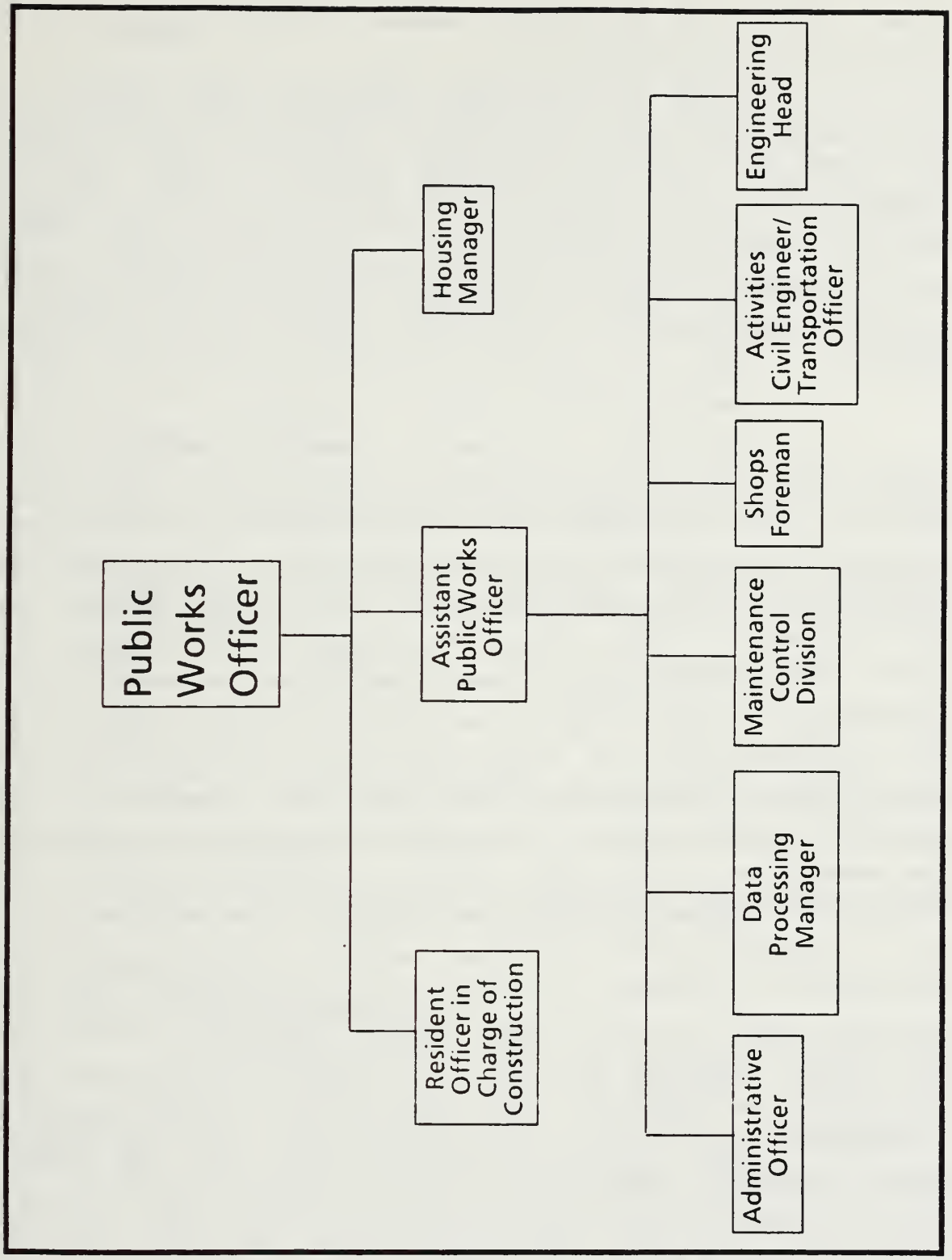


Figure 1.2 Public Works Department Organization

money provided by the tenant command to Public Works at the beginning of the fiscal year.

It is of tantamount importance for the PWO to know the dollar value of resources consumed to date by each LMC and the value of resources charged to each SEG for reimbursable jobs in the fiscal year to ensure that appropriate resources are maintained and managed efficiently, and that reasonable policy decisions are made concerning the priority assigned to reimbursable versus non-reimbursable jobs.

c. Comptroller

The Naval Postgraduate School Comptroller is tasked with monitoring the use of labor funds. The Comptroller keeps the official labor statistics, determines policies such as that governing the assignment of appropriate acceleration rates to be applied to various labor charges and must account for any differences between hours reported on labor distribution cards (i.e. those used to ensure that particular appropriations are spent in areas for which they were appropriated) and hours reported on time cards (i.e. those used in processing the payroll). The Comptroller also must have access to valid information to effectively monitor the execution of the labor budget. Although in theory the Comptroller Office maintains the official labor figures, they do so on WANG office equipment which currently lacks the capability to transfer the data to the Authorized Accounting Activity (AAA) at Naval Supply Center, Oakland. Therefore, the actual timecards are mailed to Oakland where they are keypunched for entry via card reader into a system known as Integrated Disbursing and Accounting (IDA). The WANG system holds Memorandum Labor Records which while not official are used to identify labor card data erroneously input into IDA and to reconcile differences between time card and labor card entries.

B. PURPOSE

The purpose of this thesis is to analyze, design and implement a prototype system able to provide year-to-date labor information with sufficient detail and accuracy to enable the PWO to base policy decisions on the information provided by the system. As a secondary objective, the thesis will address the larger problem of integrating some of the systems used by other entities to track the same data and will recommend a course of action to be taken to eliminate unnecessary redundancy of data input.

C. RESEARCH QUESTIONS

What data elements would have to be included in a labor tracking system which would satisfy both Public Works Department and Comptroller requirements?

How should these data elements be organized to minimize insertion and deletion anomalies without sacrificing acceptable retrieval times on Government-owned microcomputer equipment?

What is the impact of a Decision Support System on the efficiency and effectiveness of the PWO?

What long term plans should be made to reduce data redundancy?

Why have previous attempts to alleviate labor card problems by computer systems been less than satisfactory?

What steps should be taken to minimize errors introduced into the system?

What software should be utilized to enhance any future attempts to integrate the various systems?

D. METHODOLOGY

1. Problem Definition

Interviews with key personnel from the Public Works and Comptroller Departments were assembled, condensed and reworded as a problem definition including a statement of scope and objectives which was reviewed, edited and agreed upon by each office. A simple prototype microcomputer system using flexible fourth-generation software was promptly introduced to assist with the proper identification of requirements.

2. **Feasibility**

The feasibility of the proposed system was evaluated. Previous attempts to automate the labor tracking function were examined. Interviews were conducted to determine shortcomings of the previous systems. Differences among the various programs were highlighted to find possible causal relationships and to illustrate the technical, economic and political feasibility of the proposed system.

3. **Analysis**

The current systems used to track NPS labor spending was analyzed in depth. A physical model of the present labor tracking systems was made. From the physical model and some additional information gleaned in interviews of personnel, a more optimal logical model was derived. This logical model, expressed by means of a data flow diagram and data dictionary, was the foundation for design.

4. **System Design**

During system design, several high-level physical alternative solutions were considered. This high-level design was motivated by a search for a general method which would provide a reasonable solution to the defined problem. A system flow diagram with perceived costs and benefits of each system was constructed.

5. **Detailed Design**

A structure chart and mini-specs were generated from the system flow analysis. The prototype was improved module by module as coding for each became complete. Testing of completed modules was performed concurrently with development of new modules. This was done to ensure that a complete working system would result from this thesis research, even if the entire requirement was not fully satisfied by the end product.

6. Implementation and Testing

The most extensive testing of new modules was performed by the Public Works Department Administration Section, with reports and results being fed to the Public Works Administrative Officer as well as Comptroller Personnel.

7. Conclusions

The thesis culminates with an evaluation and description of the status of the developing system with specific recommendations relative to its future refinement.

E. EXECUTIVE SUMMARY

Employees of the Public Works Department at NPS, the NPS Comptroller and contract personnel in Oakland manually enter employee, labor and job data into four independent computer systems. This results in excessive labor effort not only to enter redundant data, but also to reconcile discrepancies among the various databases.

The PWO does not receive information from the Comptroller in a manner timely enough to manage the Public Works payroll budget as closely as he desires. As a result, the Public Works Department uses data which they compile internally. Since most of this work is done manually, it lacks the reporting flexibility and the data integrity necessary to provide sufficient support to the PWO. Because of the methodology employed in the current system, errors are frequent and difficult to identify.

The goal of this thesis is to develop a labor cost tracking system for the PWO which has sufficient flexibility to be adjusted as additional user requirements evolve. The prototype was implemented concurrently with the present system and will continue to run concurrently with it until complete confidence in the automated system is gained. The cost of the system is under $3500. The Public Works Data Processing Manager estimates direct local savings resulting

from the reduction of data entry redundancy to be at least $7500 annually. It is difficult to quantify additional savings which will result from more reliable payroll decision support provided to the PWO. It is reasonable to conclude that the system is likely to pay for itself in approximately six months of implementation.

II. PROBLEM DEFINITION

A. LABOR DATA INTEGRITY

Since the purpose of this thesis is to provide the PWO a tool for monitoring funds status by LMC and SEG, and since this tool may be used to influence policy decisions, it is of primary importance to insure the accuracy of the data entered into the database. In this regard, due consideration must be given to the types of errors which are most likely to occur and to the means by which their occurrence could be minimized.

1. Source Errors

More often than might be expected, incorrect Social Security Numbers, Job Numbers or hours worked entries appear on the labor cards turned in by employees. To expect an entry clerk to check each entry of some eighty-five labor cards submitted each day would be unreasonable. Ideally, if the computer could have a set of rules which would allow it to identify timecards which appear to be incorrect, many of these errors could be caught. If each Social Security Number was sought in a file of current employees and each Job Number was sought in a file of current jobs, bogus SSN's and Job Numbers could be identified and corrected prior to the addition of the suspected record to the memorandum file.

This system would not be fail-safe in that if an incorrect but currently active job number were entered, it would not be rejected. If a valid SSN were entered, however, a suspected error reading would be generated with the daily report when the program would realize that greater than eight regular hours had been entered for an employee on one day. This system also promises to be helpful in identifying typing errors on entry prior to any corruption of the database. Preliminary experience with the prototype indicates that this

error-checking system is highly effective in increasing labor card data integrity.

2. Keypunch Errors

Errors are currently introduced into the system when employee labor card data is incorrectly keypunched. The Public Works Data Processing Manager estimates at least two man-days per month are required to find and correct errors introduced by keypunching alone.

This type of error could be eliminated if the Public Works memorandum records already existing in machine-readable form were to be transmitted to Oakland. It would seem that the most reasonable transfer could take place via telecommunications, but even transfer of magnetic media would eliminate this type of error.

B. EXISTING CAPABILITIES

1. Base Engineering Support, Technical (BEST)

The Base Engineering Support, Technical (BEST) system was installed to provide the Public Works Department with information support in the areas of facilities maintenance, utilities, transportation, and family housing. The maintenance function includes modules which assist the PWO in evaluating the effectiveness of various cost centers. More specifically, these modules indicate how a cost center or even specific employees are performing with respect to Engineered Performance Standards (EPS). BEST also evaluates the accuracy of cost estimates.

To provide these functions, BEST requires the entry of actual job data in a format different from that used on the labor cards. Hence, this information is entered into BEST separately at the shops level.

BEST was not initially designed to support the PWO in budget execution. The module which would make this possible was not included in BEST's initial development because a similar function was to be included in the BASIS financial

information system expected to be operational in the mid 1990's. Currently, the actual hours expended on a given job or job category as per BEST is reported in a separate system without significant error checking. No convenient mechanism exists to allow legitimate comparisons between total effort as per BEST and total effort as per the labor cards. The level of confidence in the actual hours expended on a job as per BEST must be significantly improved. The PWO would like to integrate the BEST labor input with the labor card input and thereby improve the integrity of the BEST system and eliminate redundancy.

2. Turbo Pascal System

In June of 1986, a Public Works employee developed a labor distribution and accounting system in Turbo Pascal for an IBM-PC. The design of the data structures for the system showed insight into the complexity of the problem. The system established a text file for employees, labor cards and jobs.

Although the system represented a significant improvement over the completely manual system, it was deficient in several key areas. The text files were manipulated by the Wordstar word processing program. This necessitated the training of entry clerks in the usage of Wordstar. Input files had to be formatted in a strict fashion. Stray characters caused frequent program malfunctions. Because the files eventually grew to be large, it was difficult to find and change entries, hence, duplicate entries and omissions were common and usually progressed through the system undetected. Because of the way the databases were joined, the program took several hours to generate the equivalent of a Fund Code Report. Documentation of the program was sparse (only three pages), and procedures were cumbersome, making the code difficult to maintain, especially after the employee left for another job.

3. AIMS System

The Comptroller Department maintains their memorandum accounting records on a WANG computer using the AIMS software package. AIMS is a relatively user-friendly off-the-shelf database program. Transactions are entered into the AIMS system when four Comptroller Department entry clerks copy data from labor cards filled out by NPS employees assigned to departments other than Public Works. The timecards are then sent to Oakland for keypunching and entry into the IDA system.

Each of the local records is deleted from the WANG system when the record appears on the IDA Transaction Listing. The reconciliation process is very time and labor intensive and would be unnecessary if the transaction data entered at the local level were transferred to the IDA system in machine readable form. The current capabilities of the WANG system preclude this alternative.

Although the AIMS system supports the Comptroller to some degree, it provides no support for Public Works.

4. LABORMON

The LABORMON System was developed by thesis students Donald H. Hildebrand, Jr. and Andrew Marafino, Jr. in early 1987. LABORMON is a system based on Lotus 1-2-3 spreadsheet macros and templates which was designed to help lower-level managers manage their payroll. It suffered from the fact that 1-2-3 does not support the relational model. Hence, the software could not support very sophisticated relations and could not easily be altered to meet long-term Comptroller requirements. The program employed very little error checking, was never implemented and was abandoned by the Comptroller Department because of an absence of top-management interest and maintenance programming support.

C. PROBLEM STATEMENT

The Naval Postgraduate School Public Works Department must monitor labor hours charged by employees to assist in managing the payroll and executing the budget. The processes of entering data into a local system, reconciling locally kept records with official records, and transferring data to official systems are too expensive.

The absence of significant error checking and an unfriendly user interface is blamed in part for the errors which are often introduced into the memorandum records. These errors sometimes propagate into the official records necessitating tedious activities of search and correction. There is a lack of confidence in the accuracy of current labor statistics as kept in the memorandum records which limits the level of support the system is able to provide.

III. ANALYSIS

A. FEASIBILITY

1. Technical Feasibility

The technical feasibility of a computerized labor tracking system is readily established by considering similar applications. Accounting applications were some of the first historically to be automated successfully. Indeed, at least four automated systems exist now which attempt to perform the labor tracking function. Each was designed by a user to do the tracking function in addition to other specific functions which varied by user. Each system was developed independently to solve a particular need without much thought to integration with other systems. The major shortcomings of the present systems have already been discussed. None of those shortcomings appear to be beyond the realm of what is technologically possible. The absence of maintenance support for programs previously developed can be corrected in the present case by assigning maintenance responsibility for this program to the Data Processing Manager in the Public Works Department. The primary concern of this thesis is to improve the integrity of the PWO memorandum accounts. Subsequent actions should involve integrating the memorandum accounts with the labor data input into the IDA system and with BEST. Conversations with project managers for BEST and data processing personnel in Oakland indicate that both systems are capable of reading ASCII files via telecommunications. Hence, the project is technically feasible.

2. Economic Feasibility

The Data Processing Manager for Public Works estimates that monthly clerical effort in entering labor card data into the computer system will save approximately 75 person-hours per month. At a cost of $10 per person-hour,

this represents a savings of $750 per month. The cost of a Zenith-248, software, and modem at GSA prices is approximately $3500. The Public Works Department employs temporary personnel to assist with clerical work. Hence, the system will pay for itself in six months in clerical savings alone. It is difficult to determine the dollar value which should be assigned to the fact that the PWO will have more accurate data on which to base policy decisions. As is often the case in information systems, the chief benefits of this system may well be intangible.

3. Political Feasibility

To be complete, political feasibility has to be addressed at two levels. First, the political feasibility of the module for Public Works memorandum accounts should be examined. The political feasibility of a more ambitious integration of programs is more difficult to ascertain, but should be evaluated.

The political feasibility of developing a module for Public Works memorandum accounts is established. The PWO requested that such work be undertaken. The permanent clerical staff is very supportive of the change. The only possible political problem for the development of this system is the possible objection of the Comptroller if he perceives direct data entry from Public Works to the AAA as a threat to a part of his mission. In that event, a requirement that the information be channeled through the Comptroller's Office rather than being sent directly to Oakland should calm any Comptroller fears of Public Works encroachment.

The political feasibility of integrating the local memorandum accounts system with the BEST system and the IDA system is much less likely. The problems of assigning responsibilities for maintaining the system, identifying additional appropriate data elements to be included and updating the system. There is no single individual or

command under the present structure which can take the overall responsibility for implementing and maintaining such a broad system which influences such a wide variety of people and commands. Though it may be politically infeasible, a single integrated or distributed database which could be accessed by each entity having a need for access would be the most efficient and cost-effective solution.

B. ALTERNATIVES CONSIDERED

1. Formal Alteration of BEST

Keep the present system and request that the Civil Engineer Support Office (CESO) in Port Hueneme alter BEST to more completely support the labor distribution function. The advantage of this is that no direct investment is involved. The disadvantage is that this action does nothing to help the duplication of effort problem in the short term. It is also questionable as to how responsive CESO would be to such a request given that the integration of BEST into a larger BASIS system which would include the financial functions is not expected to be on-line until the mid 1990's.

2. Stand-alone System

Develop a comprehensive single-user microcomputer based prototype system which automates the keeping of Public Works memorandum records for labor tracking purposes and which can share data with BEST via floppy disk. Under this alternative, arrangements could be made with Naval Finance Center, Oakland to send the labor data via telephone modem in standard ASCII format for use in data entry to IDA. This would eliminate costs associated with shipping the labor cards, keypunching the data and identifying discrepancies. This system would utilize one of the Zenith-248's recently acquired by Public Works. The cost of the system would be extremely small (about $3500) and if the PWO finds the system to meet his needs, a copy could be sent to CESO with a

request that future versions of the BEST system incorporate the salient features of the prototype.

Since a subset of BEST is currently being sized down to microcomputers for smaller installations which do not need or cannot afford the larger Honeywell system, the prototype functions might be integrated into the downsized system. If the module was well received, additional integration with the larger Honeywell-based BEST system would be justified.

C. RECOMMENDED ALTERNATIVE

Since the actual labor data is identical to that which is needed by Public Works to create the local memorandum records and the labor distribution report, alternative 2 is recommended.

1. Stand-alone Advantages

Actual employee, labor card and jobs data should be entered into a Zenith-248 directly from employee and job records and each employee's daily labor distribution card. The microcomputer could manipulate and transfer the data in machine readable form to IDA for labor distribution purposes. The source code would be relatively easy to maintain. Future enhancements such as automatic generation of overtime reports, other standard queries of the database and communication with BEST to support statistical analysis using Engineered Performance Standards could be added to the system by the Public Works Data Processing Manager.

2. Advantages as a BEST Standard

Should a future version of this system be accepted as a standard for BEST, most of the advantages above would accrue. The disadvantage of the system being somewhat less flexible would be more than compensated for by the transparent integration of the data into the BEST database. The system would then be standardized and available at all BEST locations reducing the need for displaced Public Works employees to learn new stand-alone procedures at every site.

3. Advantages as an IDA Standard

If this system were expanded to support Comptroller requirements, its introduction to lower echelons of command would be consistent with recent initiatives to make lower level managers more responsible in managing the payroll. This system has the potential to serve not just Public Works, but other sub-activity groups also. Further research is necessary, especially concerning the concept of audit trails as well as specific additional information support which may be required by other subactivity groups before this particular standard could be defended.

D. LOGICAL MODEL OF SYSTEM

1. Data Flows

Figure 3.1 illustrates the data flow of the system used by Public Works prior to the introduction of this prototype to maintain their local memorandum records. Figure 3.2 shows the data flow of the new system as implemented with this prototype. Figure 3.3 is the first decomposition of the data flow of the new system. The system has five major inputs and produces four major outputs.

2. Major Inputs

The major inputs include the job order information from the public works accounting technicians, the completed labor cards from the shops, civilian employee information from the Civilian Personnel Office (CPO) and a transaction listing and discrepancy report from the NPS Comptroller. A more rigorous explanation of the processes of the new system is found in the section covering program structure where each module will be discussed in some detail. Additional details can be gathered by examining the source code in Appendix C.

Job order information from the public works accounting technicians is generated whenever a new job order number is authorized or whenever a completed job is closed out. A copy of the new job authorization or closeout is sent to the

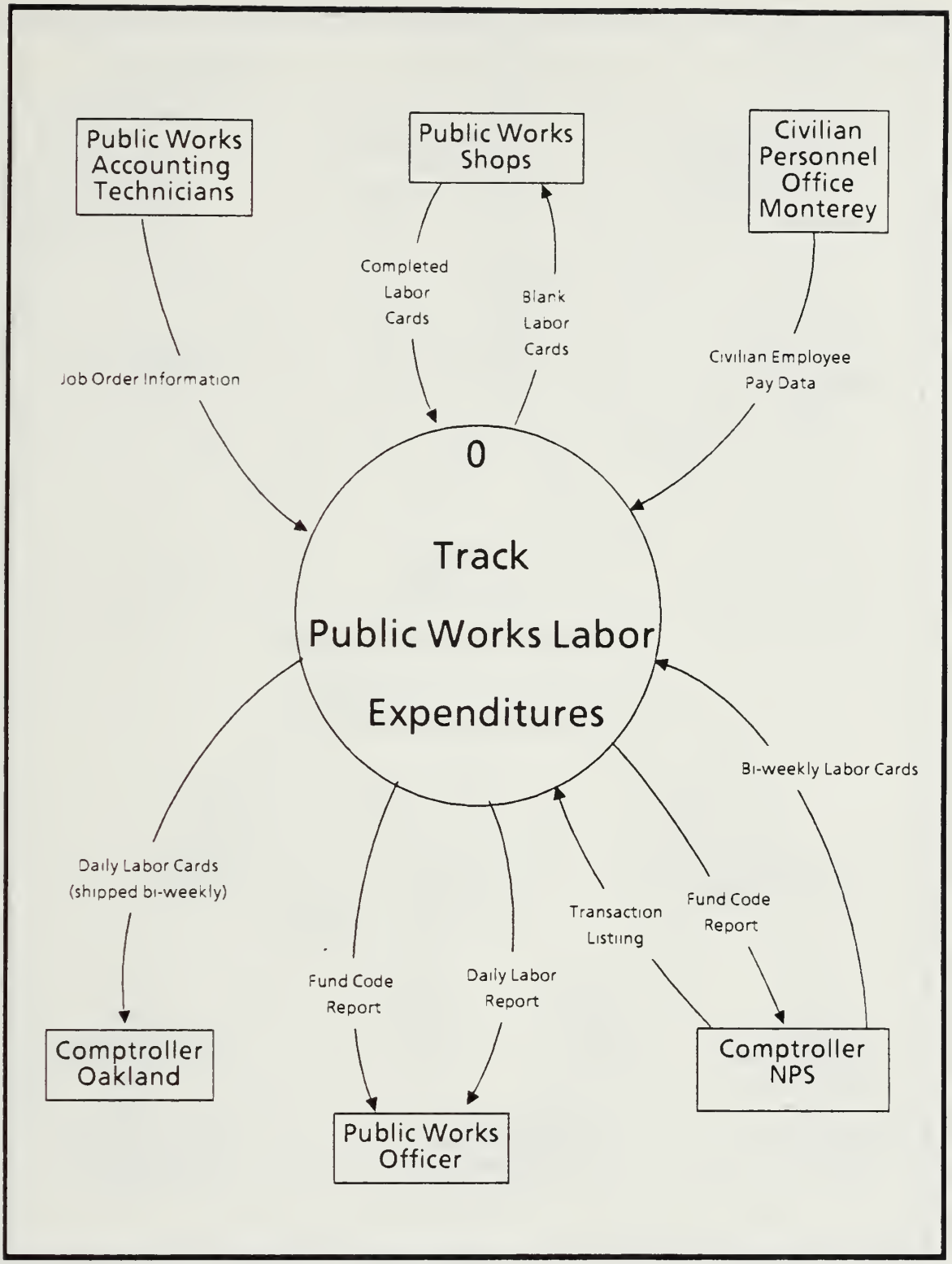


Figure 3.1 A Top-Level
Data Flow of the Previous System

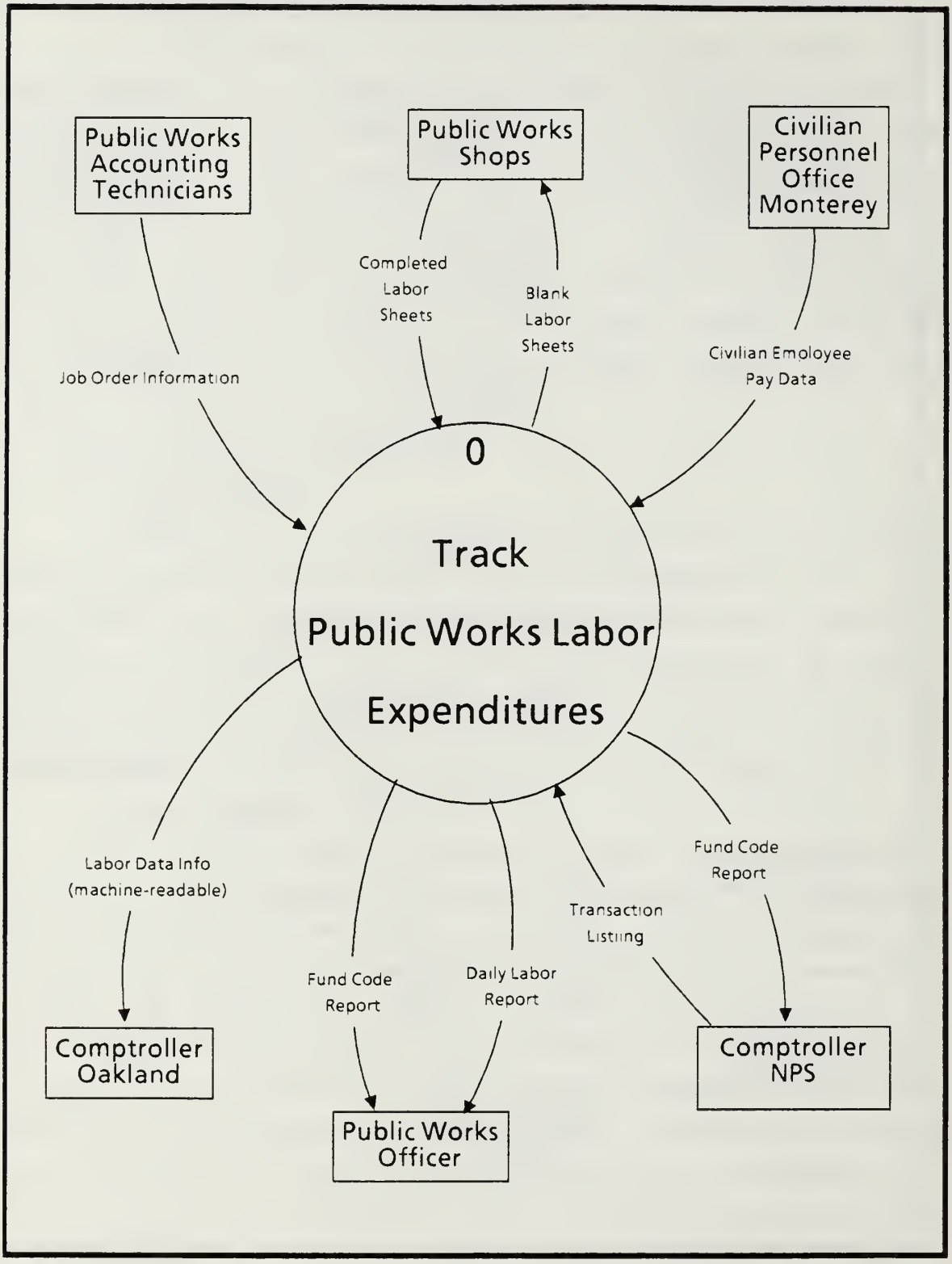


Figure 3.2 A Top-Level
Data Flow of the New System

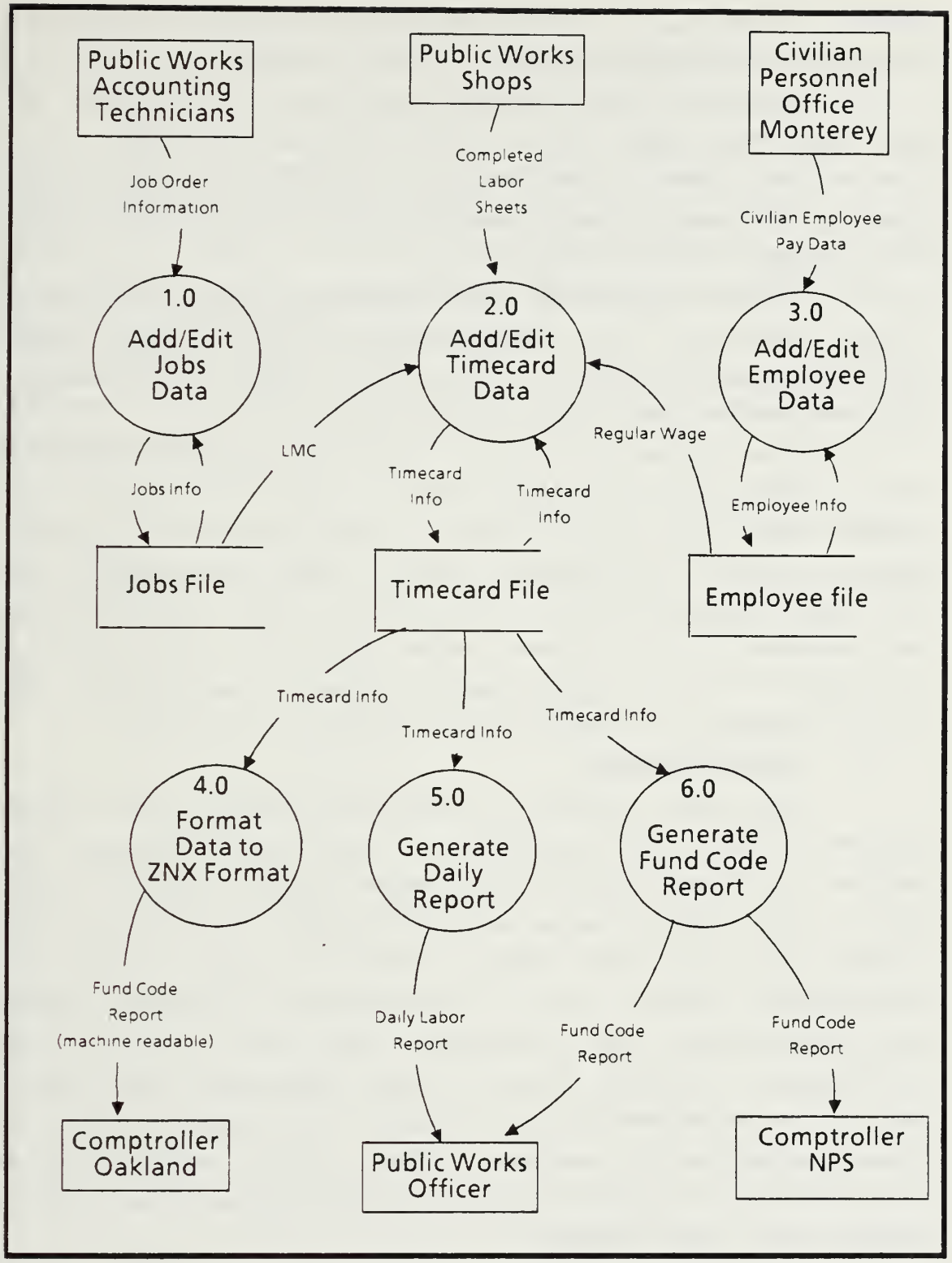


Figure 3.3 A Decomposed
Data Flow of the New System

to the entry clerk. This information is used to trigger the entry or deletion of records to the jobs file.

A daily labor distribution card is filled out by or for each wage grade Public Works employee. Entries into the system from these cards include date worked, Social Security number, and the number of regular and overtime hours charged to each job worked that day.

Civilian employee data including notification of new hires, promotions, pay adjustments and the like are reported to the system from CPO. A copy of all official documents of this nature is sent to the entry clerk.

The transaction listing and discrepancy reports are reports which AAA in Oakland sends to the NPS Comptroller who passes them on to Public Works. The transaction listing is used to check for keypunch errors. The discrepancy report indicates inconsistencies between the number of hours worked as reported by the labor cards and the number of hours reported on the timecards used to compute the payroll. This report can help to locate source errors in the data.

3. **Major Outputs**

The major output of the Public Works Labor Tracking System is the Fund Code Report. The Fund Code Report is a series of three reports which sums regular hours, overtime hours, total hours unaccelerated regular, overtime and total charges per job and groups totals by LMC for all non-reimbursable jobs. It also calculates accelerated charges for reimbursable jobs and groups totals by LMC and by SEG. Reimbursable charges against jobs associated with certain LMC's are accelerated by a higher acceleration rate than other reimbursable charges. The program allows for the acceleration rates and the set of LMC's associated with the higher rate to be changed.

The Fund Code Report is run every two weeks. It can include hours and charges assessed between two arbitrary

dates for which labor card data is contained in the database. This report is used by the PWO to set labor policy and by the NPS Comptroller to monitor the memorandum labor accounts.

The Daily Labor Report is a brief echoing of the day's labor card entries. An exception report is run prior to the generation of the daily report to identify abnormal entries for use in correcting any erroneous source data before such data is promulgated. The daily report lists and totals the day's hours and charges.

Other reports available include an alphabetical employee listing and a listing of active jobs with associated LMC's and SEG's.

IV. DESIGN

A. DATA STRUCTURES

The small number of data elements used in developing the desired program are deceiving in that complex relationships exist between many combinations of the elements. To illustrate, a logical database will be constructed in which all relations are normalized to Third Normal Form (3NF) and inherent operational problems of a normalized form as well as anomalies resulting from an unnormalized form will be considered.

1. The Employee Relation

The employee relation is composed of various data elements related to employees. The following attributes will be considered initially:

SSN	Social Security Number of employee
NAME	Name of employee
REG_WAGE	Regular hourly wage of employee
GRADE	Paygrade/step of employee
WC	Work Center to which employee is assigned. (Aids in locating employee.)
CLOCKCODE	Aids in locating employee.

Since there are no blank fields for any employee, First Normal Form (1NF) is achieved. Since SSN uniquely identifies each record in the relation, and since each attribute of the relation is fully functionally dependent on SSN, Second Normal Form (2NF) is achieved. However, REG_WAGE is functionally dependent on GRADE, hence, Third Normal Form (3NF) is not achieved because of transitive dependence. To achieve 3NF, the attributes would have to be decomposed into two files as follows:

Employee file :

SSN
NAME
GRADE
WC
CLOCKCODE

Pay table file:

GRADE
REG_WAGE

The problem becomes more complicated when we consider the requirement to be able to produce a job history. Some method of recording job history is required in this system because of the dynamic nature of employee data compared to the static nature of charges reported on a labor card. Suppose an employee earns $10 per hour on Monday, but has his wage increased to $11 per hour on Tuesday. Suppose further that this employee charged eight hours to a job on Monday and eight hours to the same job on Tuesday.

The labor tracking system, when queried about the employee's wage on Tuesday should show $11, but when queried about the employee's total charges to the particular job in question should indicate $168, $80 of which was charged on Monday and $88 of which was charged on Tuesday. The question of the manner in which a date field should be properly stored while keeping the data files in Third Normal Form is somewhat problematic from a practical point of view.

Assume that DATE is included in the list of entities to be tracked. REG_WAGE is now dependent on GRADE as well as the new element DATE. This forces the key to the Employee file to become DATE + SSN. NAME is now no longer fully functionally dependent on the key (assuming that people don't change their names). Thus, further decomposition is necessary to normalize the database structure. The functional dependency among the basic data elements is illustrated below:

```
SSN -> NAME
DATE + SSN -> GRADE
DATE + GRADE -> WAGE
```

GRADE is functionally dependent on DATE and SSN because GRADE can vary with time and worker to worker. WAGE

is functionally dependent on DATE and GRADE because federal pay tables change with time as new legislation is introduced. After decomposition, the normalized files should resemble the following: (keys are underlined)

Employee file:	Grade file:	Pay file:
SSN	SSN	DATE
NAME	DATE	GRADE
WC	GRADE	REG_WAGE
CLOCKCODE		

The practical problem of performing a retrieval under such a normalized form works against the user requirement for a quick and responsive system. Suppose labor card data exists for a given date which is to be linked to other files to retrieve the employee's regular wage (REG_WAGE) for that date. Using SSN and DATE from the labor card to key the Grade file, the appropriate GRADE can be found. Using DATE and GRADE to key the Pay file, REG_WAGE can be determined. Thus, two retrievals would be required to answer the query. The seek time required to perform such functions on a microcomputer may preclude the use of this level of decomposition in practical implementations having large volumes of data and a large incidence of this kind of query. Another problem of the above structure is that it requires one Grade file record for each employee for every day of the year on which that employee worked.

Furthermore, the Public Works Department is not concerned about entering the entire pay table each time a change occurs for their small number of employees. Neither are they generally interested in assembling a complete job history by date for each employee. For these reasons, the employee data structure used in the prototype system is simplified to encourage acceptance by Public Works as a workable memorandum system.

If this system is to be refined to become a standard, it is strongly recommended that the 3NF data structure which

includes DATE be implemented. The pay tables could be distributed via floppy diskette or on a Government on-line bulletin board to preclude each activity from having to enter the complete federal pay scales each time they change. If pay increases are across the board percentage increases and the activity had possession of dBASE-III PLUS, the activity could update a pay scale file quite easily using dBASE's "REPLACE" statement. A restructuring of the database structures could be implemented which would have little effect on the interface or workings of the prototype model.

The physical data structure implemented on the prototype is illustrated below:

Employee File:

SSN
NAME
GRADE
REG_WAGE
WC
CLOCKCODE

The anomaly associated with this form is that the regular wage (REG_WAGE) associated with a given GRADE is lost if the only Public Works employee having that GRADE is promoted or terminated. This anomaly is of little concern to Public Works as they have fewer employees than there are grades and have no real desire to reflect this relationship in their system. It was deemed in the best interest of Public Works to relinquish the absence of this anomaly for better system performance.

2. <u>The Timecard (i.e. Labor Card) Relation</u>

The labor card relation must contain the information reported on the daily labor card including employee SSN, job order numbers, regular hours worked on each job, overtime hours worked on each job, and the date. Each employee can charge hours to one or to many job numbers. The total of all regular hours charged per day cannot exceed eight. In the logical model, the relation resembles the following:

Timecard File:

DATE
SSN
JOBNO (Job Number)
REG_HRS (Reg hours charged this DATE by SSN to JOBNO)
OTHRS (Overtime hours)

The physical model differs from the logical model in that the LMC, SEG, REG_WAGE, and OT_WAGE fields are added to each record. This violation of normalization is justified in the prototype in part because the design of the employee relation does not include a job history as previously discussed. The history is kept in the timecard file so that employee wage changes and terminations do not affect queries involving jobs worked at the previous wage or by the former employee.

Because overtime wage is not simply 1.5 times regular wage in all cases, and because an employee's overtime wage can vary drastically as a result of a temporary detail to another job, the program determines the appropriate overtime wage for the date the labor card was entered and saves it to the timecard file. The other reason these fields are included in defiance of normalization rules concerns the unacceptable speed at which dBASE-III PLUS (even when compiled with Clipper) executes a join command on large databases. For each record of the first of two databases to be joined, the program sequentially retrieves each record of the other database as part of the process. Hence, if the first database has M records and the second has N, the performance is on the order of M times N. This system, were it to use joins to bring all relevant data into one file, would use two separate joins. With E records in the employee file, T records in the timecard file and J records in the jobs file, the performance would deteriorate to a sluggish order of the sum of E times T and T times J.

Conversely, with these fields permanently in the timecard file, no joins are necessary. Since the employee

and job files are indexed and must be queried to verify the validity of data input, it is much easier and less time consuming to program the join routine as part of the data input routine. Thus, wage rates and LMC's are added to the timecard file during input although this process is transparent to the user. This scheme also enables the built-in dBASE report generator to be used as an ad hoc query tool as most relevant data is in one file with no need for further file manipulation. It also allows indexes using the SEG or LMC attributes to be engaged in a more straightforward manner. (i.e. No index is required on a field in a database linked by a SET RELATION command.) Disadvantages of this approach include an increase in the required amount of secondary storage space and the introduction of contradictory information in the timecard SEG and LMC fields if a change is made in the Jobs file.

3. Jobs File

The Jobs file is consulted during labor card entry to ensure that a current job order exists for the job number being entered, and to identify the associated LMC and SEG. Thus, the file structure is as follows:

Jobs File:

JOBNO
LMC
SEG

B. PROGRAM STRUCTURES

The prototype was assembled in a modular structure with extensive parameter passing to support a low degree of coupling between highly cohesive modules to facilitate further program refinement and maintenance. Figure 4.1 illustrates the inter-module relationship. Each module is briefly described by function below.

Additional details can be ascertained from examining the commented source code included as Appendix C.

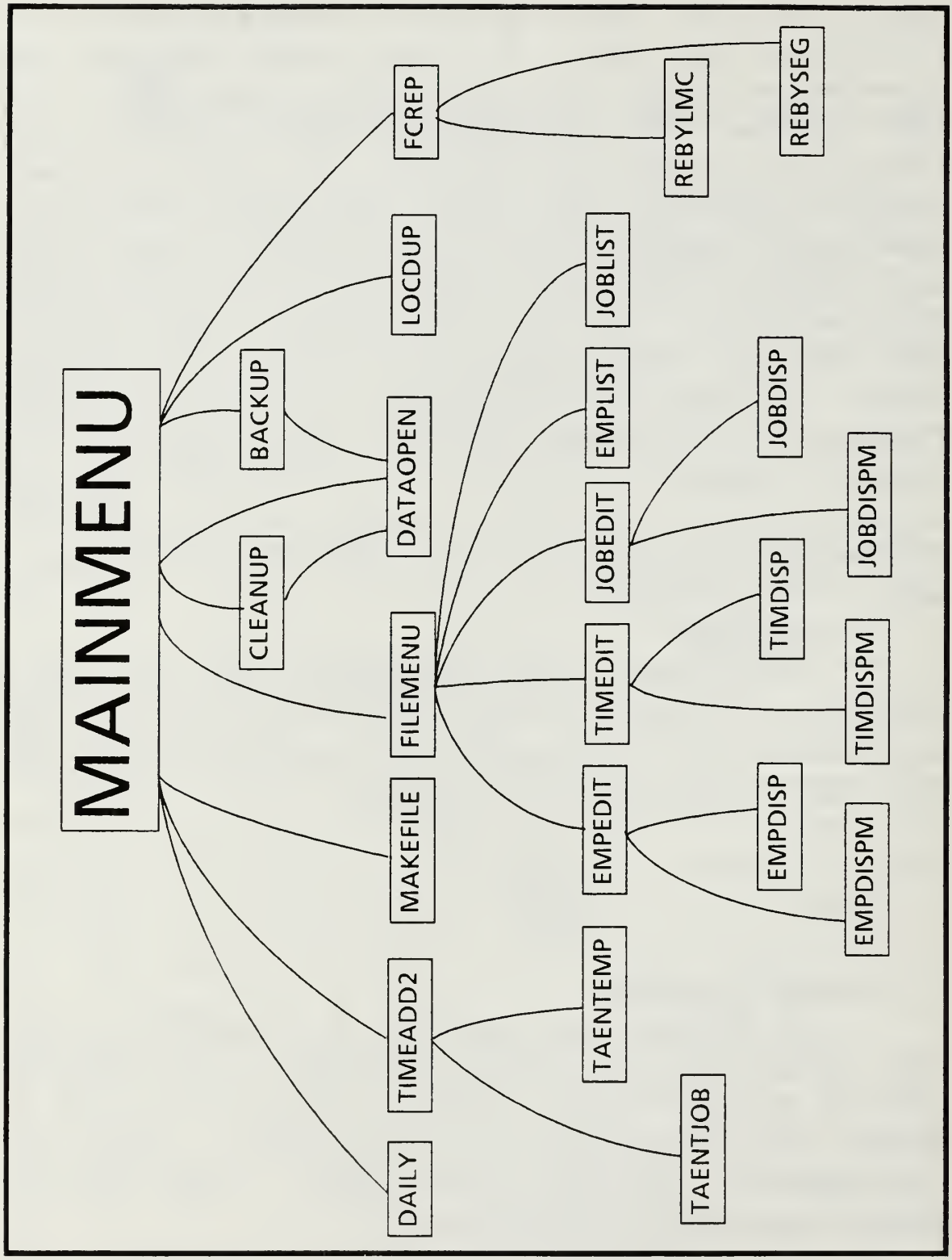


Figure 4.1 Structure Chart

1. Control Structures

It was determined that the new system would have to be easy to use to be accepted. Thus, an entirely menu-driven system was conceived. The control structure to support this philosophy uses a program module to control each full-screen menu by displaying menu choices, obtaining user preferences and calling appropriate submodules. Full-screen menus are used to select from among various functions. In contrast, line menus which generally appear on screen line 23 are used to obtain information relative to a particular intra-module function and are generated by the module corresponding to the function being performed.

Modules which generate full-screen menus include MAINMENU which generates the opening menu and FILEMENU which generates the submenu for editing files. BACKUP generates the submenu for backing up and restoring the database files and also absorbs the code for the backup functions because of its brevity. Other modules although they do not generate full-screen menus, operate as basic control modules by coordinating submodules in the execution of more complicated functions. These modules include TIMEADD2 which coordinates the entry of labor data and FCREP which coordinates the generation of the three Fund Code Reports.

2. Data Entry Structures

Because the majority of human time with the computer labor tracking system is spent entering labor card data, much thought was invested in designing the labor card entry modules to assist in early error detection and to be pleasant to use.

TIMEADD2 generates the entry screen, displays a line menu which allows the user to enter a date, Social Security Number or job number and accepts and checks the date of the labor data to be entered. TAENTEMP ensures that the Social

Security Number entered on the labor card corresponds to one in the EMPLOYEE file and fetches the corresponding regular wage rate.

After a valid Social Security Number has been entered, TAENTJOB is called to verify the presence of the job number in the JOBS file and to fetch the corresponding LMC and SEG.

3. Editing Structures

The employee file undergoes changes when an employee is added, deleted or experiences a change in pay. The EMPEDIT module provides a convenient mechanism to find a particular employee record, to skip sequentially through the file in either direction, to make changes to information stored in the file, to add new records to the file, and to delete and recall individual employee records. To accomplish its function, EMPEDIT calls EMPDISP which displays employee database values on the screen and EMPDISPM which displays memory values on the screen. Requested changes are made in memory and the user asked for confirmation prior to saving changes to the database to increase system integrity.

The jobs file is edited in much the same way by the JOBEDIT, JOBDISP and JOBDISPM modules. JOBEDIT has a trap which prohibits the user from entering two records having the same job number.

Similar functions are performed in the timecard database by the TIMEDIT, TIMDISP and TIMDISPM functions except that the addition of new records is not allowed. Adds should be performed via the TIMEADD2 module where more extensive error checking exists. Since timecard has a compound key, TIMEADD2 prompts for each portion of the key element by element to facilitate the location of a particular timecard record. Individual records can be accessed this way to correct errors prior to the printing of the daily report.

4. Report Generation Structures

The prototype system produces two major types of reports. Reports which primarily reflect database records will be identified as lists. Reports which consist primarily of information derived from one or more records in the database will be identified as derived reports.

The DAILY module produces a list which echoes labor card data corresponding to a particular day and generates an exception report of daily labor entries in which errors are suspected. The exception report is derived by sequentially visiting each record in the employee file and summing the regular and overtime hours charged on the date in question by each employee. A suspicious circumstance is identified if any of the following conditions are met:

The employee is a white-collar (GS) employee whose total regular hours charged differs from the number of regular hours in the pay period.

A GS employee charges overtime hours.

A blue-collar (WG) employee charges more or less than eight regular hours.

A WG employee charges more than two overtime hours.

When a suspicious circumstance is identified, the employee's name, Social Security Number, Grade, Work Center, and the total regular and overtime hours charged on that particular date is listed in the exception report. A listing in the exception report does not necessarily indicate that there is an error in the data. There are times for instance that GS employees work overtime. The exception report will identify employees for which no labor card was received as well as employees whose labor card data was inadvertently entered twice or who entered an incorrect number of regular hours. The report prints key information from which the labor card can be easily accessed and corrected by the TIMEDIT module. The Work Center is printed to facilitate contact with the employee for correction of errors made on the card.

The FCREP module is the control module which coordinates the generation of three derived reports for use by management. The first displays the total number of dollars charged to each non-reimbursable job and indicates total charges to each LMC. It calculates straight costs (i.e. no acceleration is added as these costs are used in managing to payroll at the PWO level).

The second report, generated by the REBYSEG module, shows the labor dollars charged to reimbursable jobs and indicates total charges to each SEG. These costs are accelerated. For reimbursable jobs having an LMC of F1C1 or F1F1, the acceleration factor is slightly higher than for reimbursable jobs assigned other LMC's. The acceleration amount added to a job is determined by totaling the hours an employee worked on the job and multiplying this sum by the employee's regular wage rate times the appropriate acceleration factor. Acceleration factors are determined by the NPS Comptroller and vary with time. As a result, the program allows the user to enter the current acceleration factors and identify the LMC's to which the higher factor applies without changing the source code.

The third report, generated by REBYLMC, reports the same accelerated labor charges for reimbursable jobs as the second report but delivers the total charges assigned to each LMC. This report is of more interest to the NPS Comptroller as it provides information useful in determining new acceleration rates.

Lists are generated by the LOCDUP, EMPLIST, and JOBLIST modules. LOCDUP searches the employee and jobs databases for duplicate entries and prints any duplicates found. LOCDUP is especially useful if some data is transferred from another machine-readable medium, as in the case of the Public Works Pascal Labor Tracking System.

EMPLIST produces an alphabetical listing of all Public Works Employees with their Social Security Numbers, regular and overtime wages, work centers, and clockcodes. The overtime wage is calculated as 1.5 times the regular wage for WG employees and the lesser of 1.5 times the regular wage or the maximum GS overtime rate for GS employees. As the maximum payable overtime to GS employees varies with time, this quantity is stored in a memory file and can be changed from the terminal by the user without changing program source code.

JOBLIST produces an alphabetical listing of all active jobs and their associated LMC's and SEG's. This list is distributed to the clerical staff for their convenience.

5. Utility Modules

Utility modules include BACKUP which facilitates the backup of the three databases, CLEANUP which removes all deleted records from each of the databases and rebuilds all the indexes and four procedures used to simplify the screen handling of all modules. The MAKEFILE module creates an ASCII file appropriate for modem transfer of data.

The L23MENU procedure receives a character variable, a set of possible responses and a prompt string. It centers the prompt string on line 23 of the screen, receives keyboard input of one character, verifies that the character received from the keyboard is contained in the set of valid characters and returns the choice character to the calling module.

The MSG procedure receives an integer line number and a character string. It centers the character string on the line corresponding to the integer line number and rings the bell.

The PAINTBOX procedure draws a box as wide as the screen from the line given by the "top" input to the line given by the "bottom" input and centers a string passed to "title" at the top of the box.

The SAYGET procedure facilitates the writing of short input routines. Often, the program needs to get a short string or number from the user. This procedure eliminates the need for calculating screen coordinates for quick interactions in which data must be input by the user.

C. SOFTWARE SELECTION

The software selected for implementing this system was the dBASE-III PLUS database package by Ashton-Tate and the Clipper dBASE compiler by Nantucket. Some reasons for these selections will be discussed below.

1. High-Level System

The original requirements for the system were largely unknown. The PWO had an idea of some of the requirements, but readily admitted that one of the purposes of developing the prototype was to clarify his needs. Further, it is unlikely that a module which satisfies the needs at Monterey will necessarily be a standard which would meet needs at other activities as successfully without some alteration. Therefore, the software must be flexible enough that the very data structures can be changed with a minimal amount of effort. The whole prototyping methodology depends on the flexibility of fourth-generation languages. dBASE-III PLUS offers a high-level flexible database environment which meets this need of a prototyping environment.

2. Native Code Advantages

The Clipper compiler for dBASE allows execution speeds over twice that of interpreted dBASE. In performing the labor card entry function in which six index files are open and three are frequently updated, the execution was significantly better when the source code was compiled.

As source dBASE code compiled with Clipper and linked with PLink 86 (included in the Clipper package) results in executable native code, multiple copies of the compiled code could be distributed freely to users without regard for

royalties or user licenses. This advantage would be particularly attractive if a refined version of this prototype were to become a standard or to be used by other Public Works Departments.

3. Integration

A concern in this era of software proliferation is the ability of a system to be integrated with other systems. Much integration of necessity is reverse-engineered after two stand-alone systems have been developed. Ashton-Tate is working on a version of dBASE which while being entirely compatible with the current version is being designed for compatibility with Standard Query Language (SQL), the standard database language for larger systems. Given the size of Ashton-Tate and the current popularity of dBASE among microcomputer users, it is reasonable to assume that software to support the integration of dBASE with other systems will become more capable and more readily available. Clipper-compiled dBASE programs can also be made to run on many local area networks (LAN's). This alternative may solve the problems of an activity too large to process all its labor data on one stand-alone system but too small to justify the purchase of a mini-computer.

4. Maintenance

The demise of many programs developed for microcomputers is directly related to the availability of programmers and analysts capable of maintaining the code. The popularity of dBASE ensures availability of maintenance programmers and tools well into the future. Indeed, the Public Works Data Processing Manager is capable of maintaining, refining and enhancing this prototype as requirements change.

D. HARDWARE CONSTRAINTS

This program was designed to run on a Zenith-248 personal computer with a 20 MB hard disk drive. This system provides sufficient storage to easily hold NPS Public Works labor data for a year. In a larger command, a greater storage capacity and/or more frequent backup of data from the timecard file may be necessary to insure proper operation of the program. The "Limitations" section of the User's Guide, included as Appendix B, discusses storage requirements for the various files.

V. IMPLEMENTATION

A. USER ENVIRONMENT

The Public Works Personnel Management System will operate at the Public Works Office in Hermann Hall. On a daily basis, an entry clerk will enters into the database system the number of hours each employee spent on each job. This information is based on employee labor cards. It is expected that no more that 350 records will be entered each day into the timecard file.

Every two weeks fund code reports are generated which compute the number of hours and total labor costs charged to each cost category. A report is run daily which echoes the day's timecard data.

The employee file remains relatively static. Edits will be performed on the employee file when an employee experiences a pay change or transfers to or from the department. The file is expected to contain approximately 150 entries at any one time.

The job order file is somewhat more active. Each time a new job order enters the system, the appropriate LMC and SEG will be entered into the jobs file. At the beginning of the new fiscal year, all previous year job orders will be closed out and new job orders entered. Records corresponding to job orders which are closed out during the fiscal year should be removed from the file at close out. The jobs file is expected to contain approximately 1000 cross references.

Because of the large daily input of data to the timecard file, the retrieval activity will be significantly lower than the update activity.

B. SECURITY ISSUES

Much of the data in the employee file is protected by the Privacy Act and should be treated as sensitive information. Current procedure calls for the computer to be kept in the Public Works Office which is locked during non-business hours. A person familiar with dBASE or with how the labor system worked could compromise the file with little difficulty. As the previous system also provided hardcopy readouts of personal data, the security of the data is not considered to be more seriously threatened now than previously. If it is desired to make the employee file more secure, software such as that included in Borland's Superkey can be used to encrypt the file when the computer is left alone and to unscramble the file prior to use. In the case of Naval Postgraduate School where the employee file can easily fit onto a floppy diskette, the file could be transferred to floppy diskette after use and stored in a more secure location.

VI. CONCLUSION

A. EVALUATION OF PROTOTYPE

The prototype currently tracks memorandum labor data for the Naval Postgraduate School Public Works Department, operating as a stand-alone system. The data entry clerks have enthusiastically welcomed the system as its extensive error-checking capabilities have detected from 20 to 30 input errors daily. When caught at this early stage, they are much less harmful as they can be quickly and easily corrected before the records are introduced to the official accounting system. The clerks have noticed that data entry is accomplished significantly faster as they can rely more on the computer to validate unusual job numbers, a tedious process formerly done manually. The user interface has also been well received, especially the labor card entry interface which allows multiple job orders to be assigned to one Social Security Number without requiring re-entry of the SSN.

The prototype if properly refined has the potential to meet other specific requirements of the NPS Comptroller, thereby replacing the AIMS system currently used for Comptroller memorandum accounts. Other lower-level managers could use the prototype to assist with their own payroll management.

The purpose of this thesis was to implement a prototype system capable of providing sufficient support to the PWO to make good policy decisions regarding payroll management in his department and to recommend a course of action leading to the eventual elimination of unnecessary redundancy in other systems. The stand-alone prototype has met the first part of the purpose. The recommendations included in the following section identify ways to eliminate redundant data entry.

B. RECOMMENDATIONS FOR REFINEMENT

1. Responsibility for Maintenance

Although the prototype system meets all stand-alone user requirements as currently defined and has been working without incident for a number of weeks, it would be foolhardy to expect the program to be maintenance-free. The major factor leading to the previous systems falling into disuse is the absence of maintenance programming performed on these systems.

Recommendation: The Data Processing Manager within the Public Works Department should be tasked with and held responsible for maintenance of, future refinement of and additions to the Labor Distribution Tracking System.

2. IDA Interface

Although the prototype has not yet successfully transferred labor card data to Oakland, a module has been included which produces an ASCII text file of the labor card information. Future versions of the program should include formatting the data into the appropriate ZNX format, once that format is defined, to facilitate the transfer of data into IDA.

Recommendation: The Public Works Data Processing Manager should Coordinate with NSC Oakland to identify the correct file format for direct transfer of labor distribution data from the local system to IDA. Establish calling procedures and communications protocols to be used. Investigate the Xmodem protocol for file transfer at 2400 baud. Eliminate the need for keypunching of labor cards. Coordinate actions with the NPS Comptroller who is responsible for the accurate posting of the official records.

The NPS Comptroller should coordinate with the PWO to obtain information on the Labor Tracking System to evaluate its potential for use in his department. If suitable, the new system should be used to transfer labor card data for

departments other than Public Works directly to Oakland via telephone modem. If major changes are required to support Comptroller functions, he should contact a technical representative for WANG and investigate the possibility of transferring labor card data from the current AIMS system.

3. Base Engineering Support, Technical

As the problem of managing to payroll is not unique to the NPS Public Works Officer, it seems reasonable to evaluate the system for possible distribution to other Public Works Departments and ultimately be incorporated as a BEST standard.

Recommendation: The Civil Engineering Support Office (CESO) in Port Hueneme should consider distributing this labor tracking system, either as a stand-alone support tool or as a prototype for a future version of BEST. Given that job estimates are already tracked in BEST, the benefits of having hard historical data to which corresponding estimates can be compared would appear to be a significant improvement in the type of support which BEST provides Public Works Officers.

4. Labor Card Elimination

Recommendation: After standard operating procedures relative to the successful electronic transfer of labor card data to Oakland are established, eliminate costly labor cards. Labor data could be input directly to a computer, on inexpensive forms designed for that purpose or on the timecards used to report hour for payroll purposes.

5. Timecard Integration

The system used to determine hours worked for pay is entirely separate from the system used to report the distribution of labor effort until the data is compared within IDA.

Recommendation: Investigate the requirements for two separate systems for reporting hours worked on jobs.

Determine whether accounting security would be compromised by integrating both systems at a lower level. Develop a system in which the employee only has to report his hourly charges once. Ensure that sufficient audit trails exist to thwart any threats of larceny or embezzlement.

6. Historical Data

Despite the successes of the prototype, there remain some functions which would be better performed if several of the databases had the capability of storing information as a function of date intervals. A date interval consists of all the dates between a "from date" and a "to date". Given the date as part of a composite key, the system would find the record corresponding to the rest of the key and having a date interval containing the desired date. For example, if the employee file could efficiently store a wage history of each employee, it would be unnecessary to store the regular wage and the overtime wage of each employee in the timecard file. Since acceleration rates and the maximum allowable GS overtime rate can change with time, it would be beneficial to store these values in a database file which made use of date intervals. The same is true of the jobs file. The implementation of this would result in a pseudo third normal form. It would not be 3NF because the same record could be accessed by many different key values, however, to decompose the file to 3NF would result in a record for each day of the year. This would seem wasteful in light of the fact that the information stored by the files in question rarely changes more than three times a year.

Recommendation: Determine if any research has been done on normalized form containing intervals. Redesign the data structures of the prototype system to conform to the pseudo third normal form resulting from the introduction of a date interval as part of a key.

APPENDIX A -- DATA DICTIONARY

DATABASE DATA DICTIONARY

FIELD NAME	DATA FILE OCCURRENCES	DESCRIPTION
ARCHIVED	TIMECARD	Boolean field which indicates whether record has been backed up
CLOCKCODE	EMPLOYEE	A number assigned to employees to assist in locating them
DATE	TIMECARD	The date on which labor charges were made
GRADE	EMPLOYEE	The civilian paygrade of an employee
JOBNO	TIMECARD JOBS	The job number
LMC	TIMECARD JOBS	Local Management Code
NAME	EMPLOYEE	Name of employee
OTHRS	TIMECARD	Number of overtime hours worked on a given date
OT_WAGE	TIMECARD	The overtime wage in effect for a particular employee on the date of the labor card
REGHRS	TIMECARD	Number of regular hours worked on a given date
REG_WAGE	EMPLOYEE TIMECARD	The regular wage in effect for a particular employee on the date of the labor card.
SEG	TIMECARD JOBS	Segment Number

FIELD NAME	DATA FILE OCCURRENCES	DESCRIPTION
SSN	EMPLOYEE TIMECARD	Social Security Number
WKCENTER	EMPLOYEE	Work Center

DATA STRUCTURES

```
Select area:  1, Database in Use: C:employee.dbf
    Master index file:  C:employee.ntx  Key: SSN
            Index file:  C:empname.ntx  Key: NAME

Structure for database: C:employee.dbf
Number of data records:     138
Date of last update    : 08/18/87
Field  Field Name  Type       Width    Dec
    1  NAME        Character     25
    2  SSN         Character      9
    3  GRADE       Character      8
    4  REG_WAGE    Numeric        6      2
    5  WKCENTER    Character      2
    6  CLOCKCODE   Character      8
** Total **                     59


Select area:  2, Database in Use: C:timecard.dbf
    Master index file:  C:timecard.ntx
                        Key: DTOC(DATE) + SSN + JOBNO
            Index file:  C:report.ntx  Key: LMC + JOBNO
            Index file:  C:report2.ntx  Key: SEG + JOBNO

Structure for database: C:timecard.dbf
Number of data records:    1407
Date of last update    : 08/18/87
Field  Field Name  Type       Width    Dec
    1  DATE        Date           8
    2  SSN         Character      9
    3  JOBNO       Character      6
    4  REGHRS      Numeric        6      2
    5  OTHRS       Numeric        6      2
    6  REG_WAGE    Numeric        6      2
    7  OT_WAGE     Numeric        6      2
    8  LMC         Character      4
    9  SEG         Character      4
   10  ARCHIVED    Logical        1
** Total **                     57
```

```
Select area:  3, Database in Use: C:jobs.dbf
    Master index file:  C:jobs.ntx  Key: JOBNO

Structure for database: C:jobs.dbf
Number of data records:     1009
Date of last update    : 08/18/87
Field  Field Name  Type       Width    Dec
    1  JOBNO       Character      6
    2  LMC         Character      4
    3  SEG         Character      4
** Total **                      15
```

MEMORY FILE STRUCTURES

TIMEMEM.MEM	TYPE	DESCRIPTION
HIACLMC	C	String of LMC's to which the higher acceleration rate applies separated by spaces
FILENAME	C	Name of daily timecard file to be transferred to Oakland
HIACRATE	N	Higher of two acceleration rates applied to reimbursable charges to jobs having LMC's contained in HIACLMC
LASTDATE	D	Most recent date of records added to the TIMECARD file
LOACRATE	N	Lower acceleration rate applied to reimbursable charges to jobs having LMC's which differ from those in HIACLMC
MAXGSOT	N	The maximum dollar value which GS employees can be paid while working overtime

APPENDIX B -- USER'S MANUAL

LIMITATIONS

This program was written for execution on a Zenith-248 with at least a 20MB hard disk. The following data file and program file storage requirements (in megabytes) are design requirements for Naval Postgraduate School in Monterey, California. Data file requirements for other locations can be estimated by applying ratios on the file storage requirements given below:

	Maximum records	Storage (MB)
Program files (EXE,FRM)	N/A	.165
Data files (DBF)		
TIMECARD.DBF	40,000	2.290
JOBS.DBF	1,200	0.018
EMPLOYEE.DBF	150	0.009
Index files (NTX)		
TIMECARD.NTX	40,000	2.213
REPORT2.NTX	40,000	1.368
REPORT.NTX	40,000	1.281
JOBS.NTX	1,200	0.035
EMPNAME.NTX	150	0.012
EMPLOYEE.NTX	150	0.008
		7.399
Backup files, sort work area		x2
		14.798

Thus, the system has ample room to function properly at Monterey on the Z-248 with a 20MB drive.

STARTUP

To install the labor tracking system, copy the following files to your hard disk:

MAINMENU.EXE
CARDLIST.FRM
EMPLOYEE.DBF
TIMECARD.DBF
JOBS.DBF
TIMEMEM.MEM

This can normally be accomplished by placing the program diskette into disk drive A: and typing copy a:*.* c: at the

"A>" prompt. Once all program files have been copied to the labor system directory in your hard drive, type "MAINMENU" <RETURN> at the "C>" prompt. Index files will be generated as needed by the program.

BEGINNING A NEW FISCAL YEAR

All job numbers change at the beginning of a new fiscal year. It is therefore suggested that a Fund Code Report be run for the previous fiscal year and that all timecard data for the fiscal year be backed up. When it is desired to clear the previous year's data from the system, the following procedures should be followed:

1. From the operating system, delete the following files:

TIMECARD.NTX
REPORT2.NTX
REPORT.NTX
JOBS.NTX

2. Copy the following files from the system diskette to the hard disk:

TIMECARD.DBF
JOBS.DBF

Procedure 1 clears the index files for all records but employee records. Procedure 2 replaces the data base files with empty structures.

FUNCTION DESCRIPTIONS

To clarify descriptions of functions in this manual, each function will be identified by name and assigned a location code consisting of a series of characters separated by periods corresponding to the menu choices which must be selected to execute the function. For example, function "Add Daily Timecard Data" is assigned location code "menu1" to indicate that the function is executed by selecting 1 from the main menu. Function "List Employee Data" is assigned location code "menu1.4" to indicate that the function can be executed by selecting 1 at the main menu then 4 at the next menu level. This system provides the facility to readily access the documentation for a given function via either referring to the index which lists the pages on which each function is discussed or by noting the keystrokes used to arrive at a menu in question. This system of documentation is attractive to users and easy for systems developers to maintain.

menu -- Main Menu

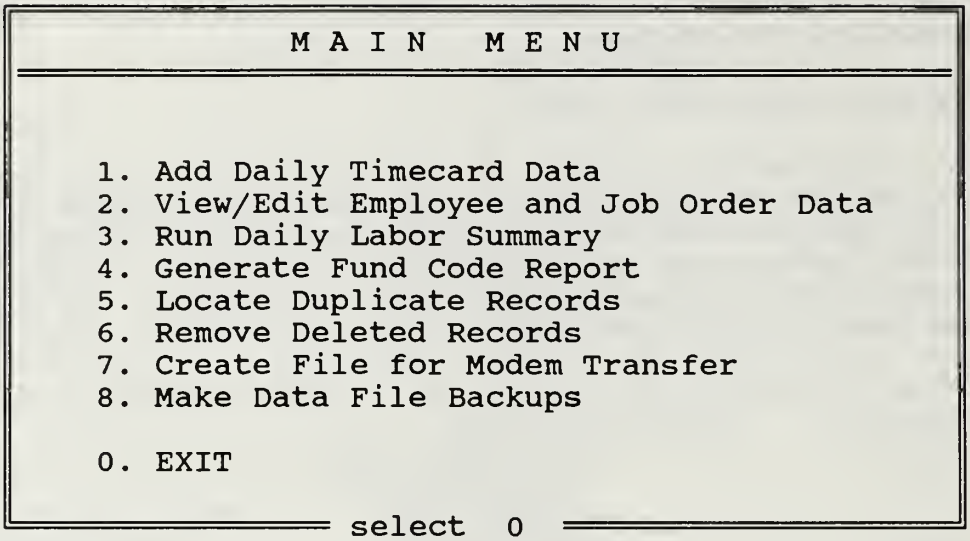


Figure B.1 -- Main Menu

menu1 -- function Add Timecard Data

This function brings the display of Figure B.2 to the screen.

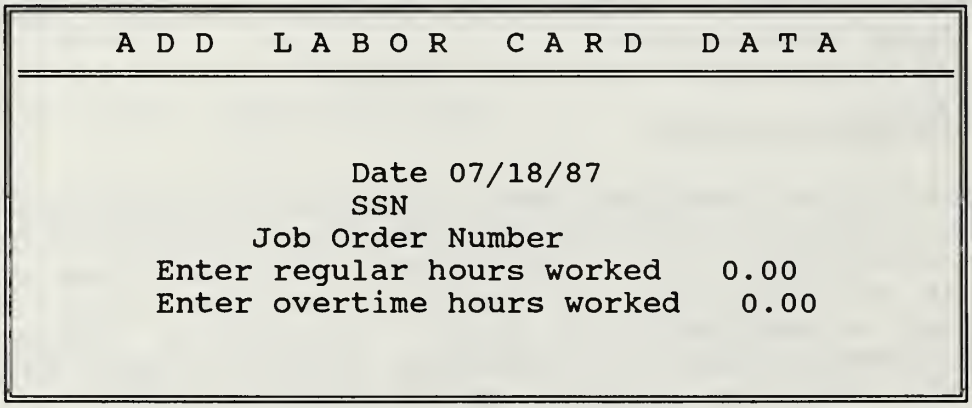


Figure B.2 -- The Add Labor Card Screen

menu1.d -- function Change Timecard Date

Upon selection of "D", the cursor moves to the "Date" field on the screen and accepts a new legal date for timecard entry. Once a date is entered, it is carried over to each

new entry until changed again with this function. This allows entry of all timecards of a given date with only one date entry. If the program is exited and re-started, the default date will be the one most recently entered prior to the exiting of the program.

menu1.s -- function Enter Timecard SSN

This function initiates the entry of a timecard into the system. Upon selection of "S", the cursor moves to the "SSN" field on the screen and accepts a nine-digit social security number. The program then checks the employee file to verify that the number entered is indeed the social security number of one of the employees. If the number is verified, the cursor moves to the "Job Order Number" field and executes the menu1.j function. If the SSN is not found in the employee file, the message "SSN not listed in employee file." appears on line 22 of the screen.

menu1.j -- function Enter Timecard Job Number

This function was introduced to preclude the entry clerk from having to type the same social security number for each job number listed on the timecard. If upon selection of "J", a validated Social Security number appears in the "SSN" field, the cursor moves to the "Job Order Number" field and accepts a job order number. The program then checks the jobs file to validate the number. If the number is found, the cursor moves sequentially to the "Enter regular hours worked" and "Enter overtime hours worked" fields to accept the number of hours charged by this employee to this job. If the program cannot find the job order number in the jobs file, the message "This job order is not in the jobs file." appears on line 22 of the screen. A menu then appears on line 23 which prompts the user to "<R>e-enter job number, <A>bandon entry." If a user attempts to enter a job number before entering a valid SSN, the message, "You must have a validated SSN before invoking this." appears on line 22 of the screen.

menu1.q -- Returns program control to main menu.

menu2 -- function View/Edit Employee and Job Order Data

This function brings the display of Figure B.3 to the screen.

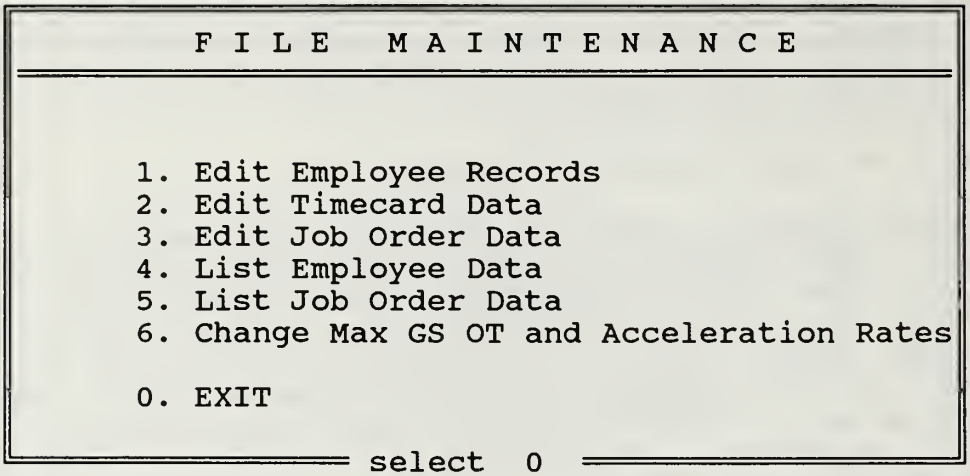


Figure B.3 -- File Maintenance Menu

menu2.1 -- function Edit Employee Records

This function brings the display of Figure B.4 to the screen.

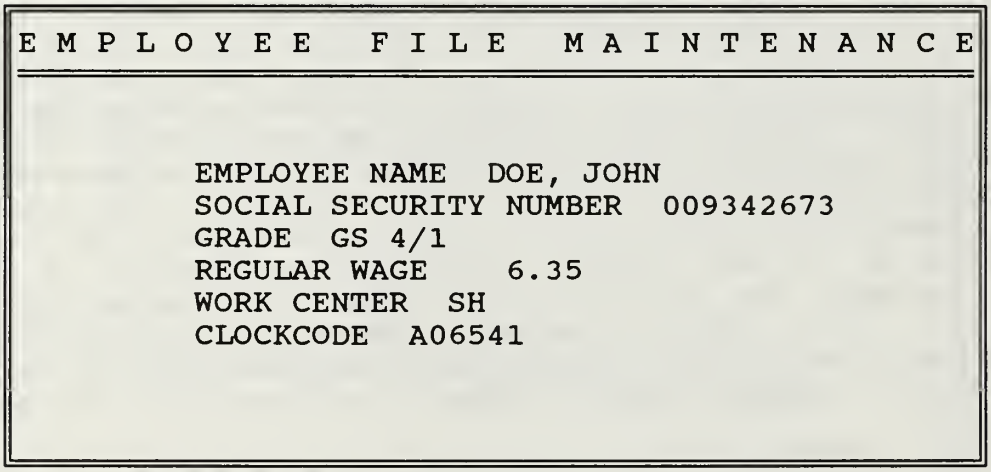


Figure B.4 -- Employee File Maintenance Menu

menu2.1.f -- function Find Employee to View/Edit

Pressing "F" causes the program to request the Social Security Number (SSN) of the desired employee. If a partial number is input (e. g. 3 <CR>), the record having the lowest

number matching the first digit(s) entered will appear on the screen. If a complete SSN is entered, the corresponding record will appear. If the SSN is not found in the employee file, the number followed by the words "not found" will appear on line 22.

menu2.1.s -- Skip to subsequent employee record and display.

If the last record of the file is shown when this command is given, the first record in the file will be displayed.

menu2.1.b -- Move back to previous record and display.

If the first record of the file is shown when this command is given, the last record in the file will be displayed.

menu2.1.c -- function Change Employee Data

This command allows the user to make changes to the employee data by moving the cursor from field to field and accepting user input from the keyboard. After all fields have been entered, the program prompts "<S>ave, <A>bort". Pressing "S" causes the changes as they appear on the screen to be entered in the database. Pressing "A" aborts the change and returns the database values to the screen.

menu2.1.a -- function Add Employee Data

This function adds an employee to the employee file. Upon pressing the "A", the program clears each field displayed on the screen and accepts new field values from the keyboard. After appropriate data has been entered into each field, the program prompts the user to "<S>ave, <A>bort." Pressing "S" results in adding the new record to the employee file. Pressing "A" aborts the operation and causes the screen to display an actual database record.

menu2.1.d -- function Delete Employee

This function marks an employee record for deletion. This record will remain in the database until it is packed. (See menu6 for more information.) Records so deleted will be indicated by the word "DELETED" appearing above the display frame to the right.

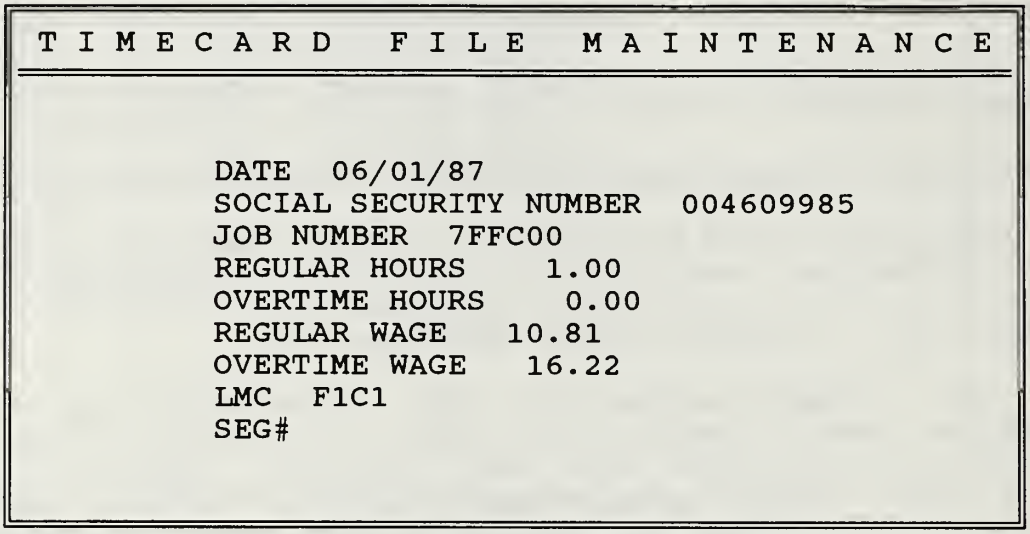
menu2.1.u -- function Undelete Employee

This function restores a deleted employee record to an undeleted status.

menu2.1.e -- returns control to main menu.

menu2.2 -- function Edit Timecard Records

This function utilizes the display screen of Figure B.5.



<F>ind <S>kip <B>ack <C>hange <D>elete <U>ndelete <E>xit

Figure B.5 Timecard File Maintenance Menu

menu2.2.f -- function Find Timecard Record to View/Edit

Pressing "F" causes the program to request the date of the timecard to find. After the date is entered, the program prompts for the Social Security Number (SSN) of the desired timecard. If a partial SSN is input (e. g. 3 <CR>), the first record of the date entered having a SSN matching the first digit(s) entered will appear on the screen. If a complete SSN is entered, the corresponding record will appear. If the date/SSN pair is not found in the timecard file, the words "Timecard not found." will appear on line 22.

menu2.2.s -- Skip to subsequent timecard record and display.

If the last record of the file is shown when this command is given, the first record in the file will be displayed.

menu2.2.b -- Move back to previous record and display.

If the first record in the file is displayed when this function is executed, the last record in the file will be displayed after execution.

menu2.2.c -- function Change Timecard Data

This command allows the user to make changes to the timecard data by moving the cursor from field to field and accepting user input from the keyboard. After all fields have been entered, the program prompts "<S>ave, <A>bort". Pressing "S" causes the changes as they appear on the screen to be entered in the database. Pressing "A" aborts the change and returns the database values to the screen.

menu2.2.d -- function Delete Timecard

This function marks a timecard record for deletion. This record will remain in the database until it is packed. (See menu6 for more information.) Records so deleted will be indicated by the word "DELETED" appearing above the display frame to the right.

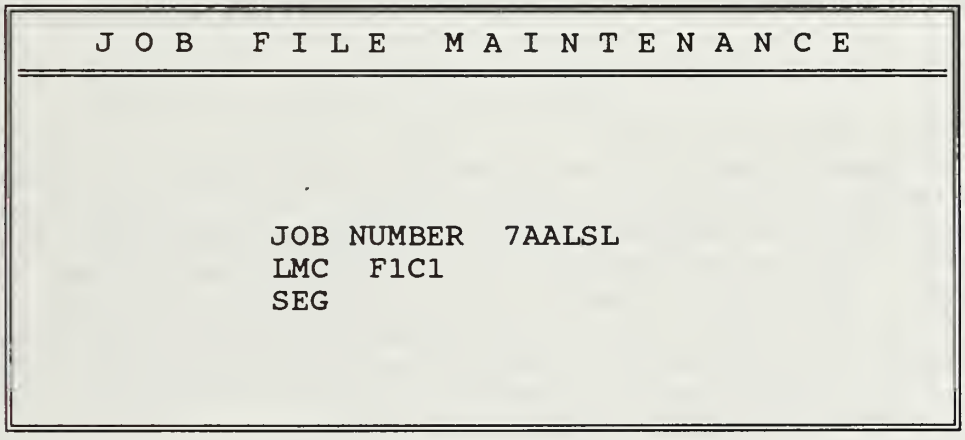
menu2.2.u -- function Undelete Timecard

This function restores a deleted timecard record to an undeleted status.

menu2.2.e -- returns control to main menu.

menu2.3 -- function Edit Job Order Data

This function utilizes the following display screen:



<F>ind <S>kip <B>ack <C>hange <A>dd <D>elete <U>ndelete <E>xit

Figure B.6 -- Job File Edit Screen

menu2.3.f -- function Find Job Number to View/Edit

Pressing "F" causes the program to request the Job Order Number of the desired job order. If a partial number is input (e. g. 7 <CR>), the first record having a job number matching the first digit(s) entered will appear on the screen. If a complete job number is entered, the corresponding record will appear. If the job number is not found in the jobs file, the number followed by the words "not found." will appear on line 22.

menu2.3.s -- Skip to subsequent employee record and display.

If the last record of the file is shown when this command is given, the first record in the file will be displayed.

menu2.3.b -- Move back to previous record and display.

If the first record of the file is shown when this command is given, the last record in the file will be displayed.

menu2.3.c -- function Change Segment or LMC

This command allows the user to make changes to the LMC or segment number assigned to a job order number. After the new LMC has been entered, the program prompts "<S>ave, <A>bort". Pressing "S" causes the job record to be entered into the database as it appears on the screen. Pressing "A" aborts the change and displays the original database values.

menu2.3.a -- function Add Job Order

This function adds a new job order to the jobs file. Upon pressing the "A", the program clears the Job Number and LMC fields displayed on the screen and accepts new field values from the keyboard. After appropriate data has been entered into each field, the program prompts the user to "<S>ave, <A>bort." Pressing "S" results in adding the new record to the jobs file. Pressing "A" aborts the operation and causes the screen to display an actual database record.

menu2.3.d -- function Delete Job Order

This function marks a job order record for deletion. This record will remain in the database until it is packed. (See menu6 for more information.) Records so deleted will be indicated by the word "DELETED" appearing above the display frame to the right.

menu2.3.u -- function Undelete Job Order

This function restores a deleted employee record to an undeleted status.

menu2.3.e -- returns control to main menu.

menu2.4 -- function List Employee Data

Upon selection of this item, a menu on line 23 prompts, "Select device: <P>rinter, <S>creen, <A>bort"

menu2.4.p -- function List Employee Data to Printer.

This selection sends the report "Employee Data" to the printer.

menu2.4.s -- function List Employee Data to Screen.

This selection sends the report "Employee Data" to the screen.

menu2.4.a -- returns to menu2

menu2.5 -- List Job Order Data

Upon selection of this item, a menu on line 23 prompts, "Select device: <P>rinter, <S>creen, <A>bort"

menu2.5.p -- function List Job Order Data to Printer.

This selection sends a cross reference of job number to LMC and segment number to the printer.

menu2.5.s -- function List Job Order Data to Screen.

This selection sends a cross reference of job number to LMC and segment number to the screen.

menu2.5.a -- returns to menu2

menu2.6 -- function Change Maximum GS Overtime Rate

This selection causes the prompts to appear on the screen as shown in Figure B.7. The current rates are displayed and new rates are accepted as input from the keyboard. The new rates are saved in a memory file upon exiting from the program and are automatically retrieved each time the program is run.

```
            Enter maximum GS overtime rate:  17.78
 Enter high reimbursable acceleration percentage: 30.9
  Enter low reimbursable acceleration percentage: 28.1
Enter LMC's to which the high acceleration percentage
applies: F1F1 F1C1
```

Figure B.7 -- Rate Adjustment Screen

menu3 -- function Run Daily Labor Summary

Upon making this selection, the program prompts for confirmation on line 23 with "Run Daily Labor Summary? (Y/N)". Responding with "N" to this prompt returns control to the main menu. Confirming this function with a "Y" response causes the program to request a date for which the Daily Labor Summary is to run by prompting on line 22 with "Enter date of daily report:". Upon entry of a valid date, the program prompts "Enter number of hours GS employees should report today:" The user responds with a <enter> if GS employees did not turn in their bi-weekly labor cards on the date appearing on line 22. If GS employees did turn in labor cards on that date, the user should enter the number of regular GS hours which should be reported for that time period (typically 80). The program then requests the user to "Select device: <P>rinter, <S>creen, <A>bort:". This refers to the output device for the listing of suspicious records.

It is convenient for the user to obtain a hardcopy listing of "suspicious" records so that any input errors on the labor cards may be corrected prior to the printing of the daily report. To determine "suspicious" records, the program goes through all records of the employee file and sums the number of regular hours charged by each employee for the day in question. The "suspicious" list contains all WG employees who charged other than eight regular hours (including those who did not turn in a card and thereby charged zero) or more than two overtime hours on the day in question. GS employees are listed if they charged a different number of regular hours than that expected from GS employees for the date in question. GS employees also make the list if they charged any overtime during the pay period.

menu4 -- function Generate Fund Code Report

Upon selection of this function, the program asks for confirmation by prompting "Generate Fund Code Report? (Y/N)" on line 23. Answering "N" here will return control to the

main menu. Answering "Y" will result in the program requesting for the first and last dates to be included in the report as follows:

```
Enter first date to be included: 07/05/87
Enter last date to be included: 07/18/87
```

The default last date is the timecard date of the record most recently added to the timecard database. The default first date is the date 13 days before the last date. Any valid date can be entered into these fields, however. If, for instance, fiscal-year-to-date totals are desired in the report, the first day of the fiscal year can be entered into the first date field and today's date can be entered into the last date field. The intent is to keep the present year's data in the database and to provide a means by which analysis of any number of arbitrary periods can be performed.

After valid first and last dates have been entered, the program prompts "Select device: <P>rinter, <S>creen, <A>bort". This allows the user to select the output device or to return to the main menu.

menu5 -- function Locate Duplicate Records

When executing this function, the program searches the employee and jobs files to find employee records having the same SSN or job records having the same job number. A printer must be connected and turned on for this function to work properly. Duplicate records are then printed out at the printer. It is strongly suggested that duplicate records be immediately corrected or deleted using the "edit function" for the appropriate database.

menu6 -- function Remove Deleted Records, Rebuild Indexes

This function "cleans up" the employee, timecard and jobs database files. A "pack" is performed which physically removes from each database all records marked for deletion. Whether or not any records are deleted, each index is rebuilt. This is valuable if records do not appear to be correctly sorted or omitted in various reports. Untimely losses of power to the system can corrupt index files. This function assumes the database file is not corrupted and rebuilds the index files. If a database file is corrupted, it is recommended that the corrupted records be deleted from dBASE by the system administrator and that this function be used to rebuild indexes.

menu7 -- function Create File for Modem Transfer

This function creates an ASCII text file containing employee Social Security numbers, job numbers, regular hours worked, and overtime hours worked for each timecard file record of a certain date. Upon selection of this function, the program prompts for the date and generates the output file in the current directory. The output file is named interactively by the user. The default name is "OUTFILE1.TXT". If no records corresponding to the entered date exist, a message stating this will appear on the screen. If an output file is named which exists, the program will request that a different name be given to the output file.

menu8 -- function Backup Data Files

The intent of this function is to assist in making routine backups of data files to floppy diskette as procedural and as painless as possible. More development on this module would be necessary for it to be a general purpose facility capable of diagnosing and correcting corrupted databases. Upon execution of this function the screen of Figure B.8 is displayed.

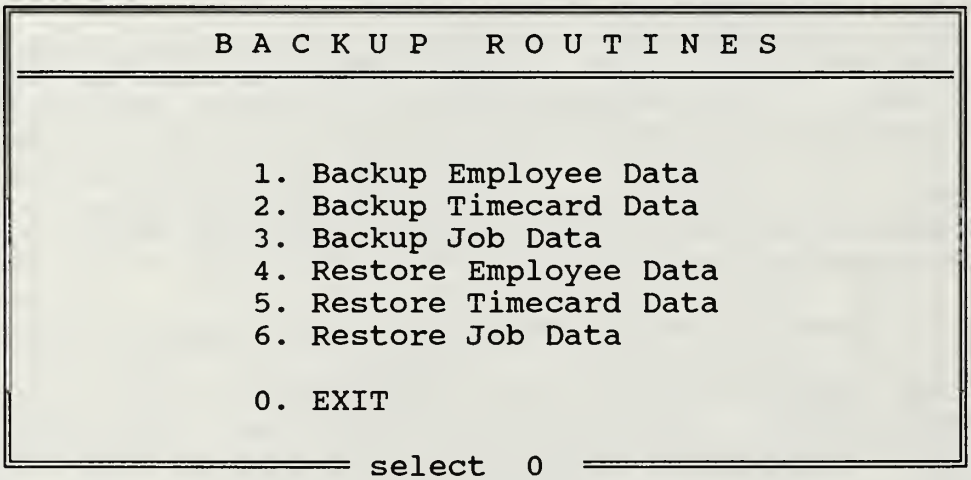


Figure B.8 Backup Routines Menu

menu8.1 -- function Backup Employee Data

Upon selection of this function, the program gives the following instructions:

```
Please insert employee backup diskette into drive A...
                <C>ontinue, <A>bort
```

If using this function for the first time, any formatted diskette will do. The assumption is that the number of employees being tracked by this system is small enough so that the file can fit on a single floppy disk. Thus, the program will overwrite an older "EMPLOYEE.DBF" file if found on the archive diskette. When a backup of the employee file is performed, the entire file is copied to diskette.

menu8.1.c -- commands the employee database file to be copied to the diskette in drive a:

menu8.1.a -- returns program control to "Backup Routines", menu8.

menu8.2 -- function Backup Timecard Data

Upon selection of this function the following instructions appear on the screen:

Please insert a formatted diskette into drive A...
<C>ontinue, <A>bort

It is assumed that because of the large size of the database, that only a fraction of the data contained therein would fit on a floppy diskette. Thus, the program will copy to drive a: only those records which have not been previously backed up. It remains the user's responsibility to ensure that backups are done with sufficient frequency to preclude the possibility of exceeding the capacity of a floppy disk with the records to be backed up.

menu8.3 -- function Backup Job Order Data

Same as function menu8.1 except that Job Order Data is being backed up instead of employee data. See menu8.1 for more specifics.

menu8.4 -- function Restore Employee Data

If the Employee Database is damaged or corrupted so severely that restoring a previously backed up database is deemed to be the best method of recovery, select this function and place the backup employee data diskette into drive a: at the prompt and press "C" to continue. ("A" aborts.)

APPENDIX C -- SOURCE CODE

```
* Program..: MAINMENU.PRG
* Author...: DAVID P. DINWIDDIE
* Date.....: 05/20/87
* Notes....:
*

SET TALK OFF
SET BELL OFF
SET SAFETY OFF
SET STATUS OFF
SET ESCAPE OFF
SET CONFIRM OFF
SET PROCEDURE TO TIMEPROC

RESTORE FROM TIMEMEM ADDITIVE

STORE ' ' TO mconfirm

DO WHILE .T.

   * ---Display menu options, centered on the screen.
   *    draw menu border and print heading
   CLEAR
   @ 2, 0 TO 18,79 DOUBLE
   @ 3,32 SAY [M A I N   M E N U]
   @ 4,1 TO 4,78 DOUBLE
   * ---display detail lines
   @  7,27 SAY [1. Add Daily Timecard Data]
   @  8,27 SAY [2. View/Edit Employee and Job Order Data]
   @  9,27 SAY [3. Run Daily Labor Summary]
   @ 10,27 SAY [4. Generate Fund Code Report]
   @ 11,27 SAY [5. Locate Duplicate Records]
   @ 12,27 SAY [6. Remove Deleted Records]
   @ 13,27 SAY [7. Create File for Modem Transfer]
   @ 14,27 SAY [8. Make Data File Backups]
   @ 16, 27 SAY '0. EXIT'
   STORE 0 TO selectnum
   @ 18,33 SAY " select     "
   @ 18,42 GET selectnum PICTURE "9" RANGE 0,8
   READ

   DO CASE
```

```
CASE selectnum = 0
   SET BELL ON
   SET TALK ON
   RELEASE selectnum, mconfirm
   SAVE TO TIMEMEM
   CLEAR ALL
   RETURN

CASE selectnum = 1
   *  DO Perform timecard append
   DO DATAOPEN
   DO TIMEADD2
   CLOSE DATABASES

CASE selectnum = 2
   *  DO Perform File Maintenance
   DO DATAOPEN
   DO FILEMENU
   CLOSE DATABASES

CASE selectnum = 3
   *  DO Run Daily Labor Summary
   DO L23MENU WITH mconfirm, 'YN',"Run Daily"+;
   " Labor Summary? (Y/N)"
   IF mconfirm = 'Y'
      DO DATAOPEN
      DO DAILY
      CLOSE DATABASES
   ENDIF

CASE selectnum = 4
   *  DO Generate Fund Code Report
   DO L23MENU WITH mconfirm, 'YN',;
   "Generate Fund Code Report? (Y/N)"
   IF mconfirm = 'Y'
      DO DATAOPEN
      DO FCREP
      CLOSE DATABASES
   ENDIF

CASE selectnum = 5
   *  Locate duplicate records
   DO L23MENU WITH mconfirm, 'YN',;
   "Locate Duplicate Records? (Y/N)"
   IF mconfirm = 'Y'
      DO DATAOPEN
      DO LOCDUP
      CLOSE DATABASES
   ENDIF
```

```
   CASE selectnum = 6
      *  Cleanup
      DO L23MENU WITH mconfirm, 'YN',"Remove "+;
      "Deleted Records and Reindex? (Y/N)"
      IF mconfirm = 'Y'
         DO DATAOPEN
         DO CLEANUP
         CLOSE DATABASES
      ENDIF

   CASE selectnum = 7
      *  DO Make transfer file
      DO MAKEFILE

   CASE selectnum = 8
      DO L23MENU WITH mconfirm, 'YN',;
      "Execute Backup Routine? (Y/N)"
      IF mconfirm = 'Y'
         DO DATAOPEN
         DO BACKUP
         CLOSE DATABASES
      ENDIF
   ENDCASE

ENDDO T
RETURN
* EOF: MAINMENU.PRG
```

```
********************************** TIMEPROC.PRG

* Procedure File..: TIMEPROC.PRG
* Author..........: DAVID P. DINWIDDIE
* Date............: 07/20/87

PROCEDURE BACKUP
STORE ' ' TO mconfirm

DO WHILE .T.

   * ---Display menu options, centered on the screen.
   *    draw menu border and print heading
   CLEAR
   @ 2, 0 TO 16,79 DOUBLE
   @ 3,26 SAY [B A C K U P   R O U T I N E S]
   @ 4,1 TO 4,78 DOUBLE
   * ---display detail lines
   @  7,28 SAY [1. Backup Employee Data]
   @  8,28 SAY [2. Backup Timecard Data]
   @  9,28 SAY [3. Backup Job Data]
   @ 10,28 SAY [4. Restore Employee Data]
   @ 11,28 SAY [5. Restore Timecard Data]
   @ 12,28 SAY [6. Restore Job Data]
   @ 14, 28 SAY '0. EXIT'
   STORE 0 TO selectnum
   @ 16,33 SAY " select     "
   @ 16,42 GET selectnum PICTURE "9" RANGE 0,6
   READ

   DO CASE
   CASE selectnum = 0
      RETURN

   CASE selectnum = 1
      *  DO Backup Employee Data
      DO MSG WITH 22, "Please insert employee "+;
      "backup diskette into drive A..."
      DO L23MENU WITH mconfirm,'AC',;
      "<C>ontinue, <A>bort"
      IF mconfirm = 'C'
         CLEAR
         DO MSG WITH 13, "Performing file backup.  "+;
         "Please wait..."
         SELECT A
         COPY TO A:EMPLOYEE
      ENDIF

   CASE selectnum = 2
```

```
   *  DO Backup Timecard Data
   DO MSG WITH 22, "Please insert a formatted "+;
   "diskette into drive A..."
   DO L23MENU WITH mconfirm,'AC',;
   "<C>ontinue, <A>bort"
   IF mconfirm = 'C'
      CLEAR
      DO MSG WITH 13, "Performing file backup.  "+;
      "Please wait..."
      SELECT B
      USE TIMECARD
      COPY TO A:TIMECARD FOR .NOT. ARCHIVED
      REPLACE ALL ARCHIVED WITH .T. FOR;
      .NOT. ARCHIVED
      USE A:TIMECARD
      REPLACE ALL ARCHIVED WITH .T.
      CLOSE DATABASES
      DO DATAOPEN
   ENDIF

CASE selectnum = 3
   *  DO Backup Job Data
   DO MSG WITH 22, "Please insert jobs "+;
   "backup diskette into drive A..."
   DO L23MENU WITH mconfirm,'AC',;
   "<C>ontinue, <A>bort"
   IF mconfirm = 'C'
      CLEAR
      DO MSG WITH 13, "Performing file backup.  "+;
      "Please wait..."
      SELECT C
      COPY TO A:JOBS
   ENDIF

CASE selectnum = 4
   *  DO Restore Employee Data
   DO MSG WITH 22, "Please insert employee "+;
   "backup diskette into drive A..."
   DO L23MENU WITH mconfirm,'AC',;
   "<C>ontinue, <A>bort"
   IF mconfirm = 'C'
      CLEAR
      DO MSG WITH 13, "Performing file "+;
      "retrieval.  Please wait..."

      SELECT A
      USE
      SELECT D
      USE A:EMPLOYEE
      COPY TO EMPLOYEE
      DELETE FILE EMPLOYEE.NTX
```

```
         DELETE FILE EMPNAME.NTX
         USE
         DO DATAOPEN
      ENDIF

   CASE selectnum = 5
      *  DO Restore Timecard Data
      DO MSG WITH 22, "Please insert archive "+;
      "diskette into drive A..."
      DO L23MENU WITH mconfirm,'AC',;
      "<C>ontinue, <A>bort"
      IF mconfirm = 'C'
         CLEAR
         DO MSG WITH 13, "Performing file "+;
         "retrieval.  Please wait..."
         SELECT D
         USE A:TIMECARD
         REPLACE ALL ARCHIVED WITH .T.
         USE
         SELECT B
         APPEND FROM A:TIMECARD
      ENDIF

   CASE selectnum = 6
      *  DO Restore Job Data
      DO MSG WITH 22, "Please insert archive "+;
      "diskette into drive A..."
      DO L23MENU WITH mconfirm,'AC',;
      "<C>ontinue, <A>bort"
      IF mconfirm = 'C'
         CLEAR
         DO MSG WITH 13, "Performing file "+;
         "retrieval.  Please wait..."
         SELECT C
         USE
         SELECT D
         USE A:JOBS
         COPY TO JOBS
         DELETE FILE JOBS.NTX
         USE
         DO DATAOPEN
      ENDIF
   ENDCASE

ENDDO T
RETURN
* EOF: BACKUP.PRG
*********************************************

* CLEANUP.PRG
* July 6, 1987
```

```
PROCEDURE CLEANUP

CLOSE DATABASES
RUN DEL *.NTX
CLEAR
? "Cleaning Employee File"
SELECT A
USE EMPLOYEE
DELETE ALL FOR SSN = ' '
PACK
? "Cleaning Timecard File"
SELECT B
USE TIMECARD
DELETE ALL FOR JOBNO = ' '
PACK

? "Cleaning Job Order File"
SELECT C
USE JOBS
DELETE ALL FOR JOBNO = ' '
PACK
DO DATAOPEN
RETURN

* eof cleanup.prg
************************************************
```

```
* daily.prg
* 7/17/87
PROCEDURE DAILY
STORE ' ' TO device
STORE LASTDATE TO TODAY

@ 22,0
@ 22,27 SAY "Enter date of daily report: ";
GET TODAY PICTURE '99/99/99'
READ
STORE DTOC(TODAY) TO CTODAY
STORE "LABOR HOURS;CHARGED &CTODAY" TO PERIOD
STORE 0 TO gshours

DO SAYGET WITH 22, "Enter number of hours GS "+;
"employees should report today:",gshours, '999.99'

DO L23MENU WITH device,'PSA',"Select device: "+;
"<P>rinter, <S>creen, <A>bort: "
IF device = 'A'
   RETURN
ENDIF
IF device = 'P'
   ? "Please be sure printer is on..."
   WAIT
   SET PRINT ON
ENDIF
?
?
? "Discrepancy Report for " + CTODAY
IF gshours <> 0
   ? "GS employees were to enter "+STR(gshours,6,2)+;
   " hours today."
ELSE
   ? "GS employees were not to be entered today."
ENDIF
?
? "The following totals appear suspicious."
?
? "            NAME                SSN     "+;
"GRADE    WC      REG       OT"
SELECT A
GO TOP
DO WHILE .NOT. EOF()
   IF SUBSTR(GRADE,1,1) = 'G'
      STORE .F. TO Iswg
   ELSE
      STORE .T. TO Iswg
   ENDIF
```

```
   SELECT B
   STORE CTODAY + EMPLOYEE -> SSN TO MCOMPARE
   STORE 0 TO MREG, MOT
   SEEK MCOMPARE
   DO WHILE DATE = TODAY .AND. SSN = EMPLOYEE -> SSN
      MREG = MREG + REGHRS
      MOT = MOT + OTHRS
      SKIP
   ENDDO

   IF (MREG <> 8 .AND. ISWG) .OR. (MOT > 2) .OR.;
   (MOT > 0 .AND. .NOT. ISWG) .OR.;
   ((MREG <> gshours) .AND. .NOT. ISWG)

      ? A -> NAME, A -> SSN, A -> GRADE,;
      A -> WKCENTER, MREG, MOT

   ENDIF

   SELECT A
   SKIP
ENDDO
IF device = 'P'
   EJECT
ENDIF
SET PRINT OFF

SELECT B
?
?
?
?
DO L23MENU WITH device, "SPA","SELECT OUTPUT DEVICE "+;
"FOR DAILY REPORT:  <S>creen, <P>rinter  OR  <A>bort"
CLEAR
IF device = 'A'
   RETURN
ENDIF

SEEK ctoday
IF FOUND()
   IF device = 'S'
      report form CARDLIST while date = TODAY HEADING PERIOD
      ?
      ?
      ?
      STORE ' ' TO wait_subst
      @ 23,0
      @ 23,0 SAY 'Press any key to continue...';
      GET wait_subst
      READ
```

```
   ELSE
      report form CARDLIST while date = TODAY HEADING ;
      PERIOD TO PRINT
   ENDIF
ELSE
   ? "No entries dated "+ctoday+;
   " were found in the database."
   WAIT
ENDIF
RETURN
* EOF DAILY.PRG
************************************************
```

```
* dataopen.PRG
PROCEDURE DATAOPEN
select a
USE EMPLOYEE
IF .NOT. FILE('EMPLOYEE.NTX')
   ? "Indexing employee file on SSN..."
   INDEX ON SSN TO EMPLOYEE
ENDIF
IF .NOT. FILE('EMPNAME.NTX')
   ? "Indexing employee file on name..."
   INDEX ON NAME TO EMPNAME
ENDIF
SET INDEX TO EMPLOYEE, EMPNAME

select b
use timecard
IF .NOT. FILE('TIMECARD.NTX')
   ? "Creating timecard index TIMECARD.NTX..."
   INDEX ON DTOC(DATE) + SSN + JOBNO TO TIMECARD
ENDIF
IF .NOT. FILE('REPORT.NTX')
   ? "Creating timecard index REPORT.NTX..."
   INDEX ON LMC + JOBNO TO REPORT
ENDIF
IF .NOT. FILE('REPORT2.NTX')
   ? "Creating timecard index REPORT2.NTX..."
   INDEX ON SEG + JOBNO TO REPORT2
ENDIF
SET INDEX TO timecard, report, report2

select c
USE JOBS
IF .NOT. FILE('JOBS.NTX')
   ? "Creating jobs file index..."
   INDEX ON JOBNO TO JOBS
ENDIF
SET INDEX TO JOBS
RETURN
* EOF DATAOPEN
**********************************************
```

```
* empdisp.prg
PROCEDURE EMPDISP
IF DELETED()
   @ 4, 50 SAY "DELETED"
ELSE
   @ 4, 50 SAY "       "
ENDIF

@ 10, 24 SAY "EMPLOYEE NAME ";
GET NAME PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!'
@ 11, 24 SAY "SOCIAL SECURITY NUMBER ";
GET SSN PICTURE '999999999'
@ 12, 24 SAY "GRADE " GET GRADE PICTURE '!!!!!!!!!'
@ 13, 24 SAY "REGULAR WAGE " ;
GET REG_WAGE PICTURE '999.99'
@ 14, 24 SAY "WORK CENTER ";
GET WKCENTER PICTURE '!!!'
@ 15, 24 SAY "CLOCKCODE ";
GET CLOCKCODE


RETURN

* EOF EMPDISP.PRG
************************************************

* empdispm.prg
PROCEDURE EMPDISPM
IF DELETED()
   @ 4, 50 SAY "DELETED"
ELSE
   @ 4, 50 SAY "       "
ENDIF

@ 10, 24 SAY "EMPLOYEE NAME ";
GET m_NAME PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!'
@ 11, 24 SAY "SOCIAL SECURITY NUMBER ";
GET m_SSN PICTURE '999999999'
@ 12, 24 SAY "GRADE ";
GET m_GRADE PICTURE '!!!!!!!!!'
@ 13, 24 SAY "REGULAR WAGE ";
GET m_REG_WAGE PICTURE '999.99'
@ 14, 24 SAY "WORK CENTER ";
GET m_WKCENTER PICTURE '!!!'
@ 15, 24 SAY "CLOCKCODE " GET m_CLOCKCOD


RETURN

* EOF EMPDISPM.PRG
************************************************
```

```
* empedit.prg
PROCEDURE EMPEDIT
CLEAR
SELECT A
GO TOP
command = ' '
opcomm = ' '

DO PAINTBOX WITH 5, 21, "E M P L O Y E E   "+;
"F I L E   M A I N T E N A N C E"

DO WHILE command <> 'E'
   DO EMPDISP
   CLEAR GETS
   DO L23MENU WITH command, 'FSBCADUE',;
   '<F>ind, <S>kip, <B>ack, <C>hange, '+;
   '<A>dd, <D>elete, <U>ndelete, <E>xit '

   DO CASE

   CASE command = 'F'
      key = '          '
      DO SAYGET WITH 23, 'Enter employee '+;
      'SSN to find:', key, '999999999'
      READ
      key = TRIM(key)
      SEEK key
      IF EOF()
         DO MSG WITH 22, '&key not found.'
         GO BOTTOM
      ENDIF

   CASE command = 'S'
      IF (EOF() .AND. BOF())
         DO MSG WITH 22, 'There are no '+;
         'records in the file.'
      ELSE
         IF EOF()
            GO TOP
         ELSE
            SKIP
         ENDIF
         IF EOF()
            GO TOP
         ENDIF
      ENDIF
```

```
CASE command = 'B'
   IF (EOF() .AND. BOF())
      DO MSG WITH 22, 'There are no '+;
      'records in the file.'
   ELSE
      IF BOF()
         GO BOTTOM
      ENDIF
      SKIP -1
      IF BOF()
         GO BOTTOM
      ENDIF
   ENDIF

CASE command = 'C'
   m_clockcod = clockcode
   m_grade    = grade
   m_name     = name
   m_ssn      = ssn
   m_wkcenter = wkcenter
   m_reg_wage = reg_wage

   DO EMPDISPM
   READ
   DO L23MENU WITH Opcomm, 'SA', "<S>ave, <A>bort"

   IF Opcomm = 'S'
      REPLACE Clockcode  WITH m_clockcod
      REPLACE Grade      WITH m_grade
      REPLACE Name       WITH m_name
      REPLACE Ssn        WITH m_ssn
      REPLACE Wkcenter   WITH m_wkcenter
      REPLACE Reg_wage   WITH m_reg_wage
   ENDIF


CASE command = 'A'
   m_clockcod = SPACE(  8)
   m_grade    = SPACE(  8)
   m_name     = SPACE( 25)
   m_ssn      = SPACE(  9)
   m_wkcenter = SPACE(  2)
   m_reg_wage = 000.00

   DO EMPDISPM
   READ

   DO L23MENU WITH Opcomm, 'SA', "<S>ave, <A>bort"
```

```
      IF Opcomm = 'S'
         APPEND BLANK
         REPLACE Clockcode  WITH m_clockcod
         REPLACE Grade      WITH m_grade
         REPLACE Name       WITH m_name
         REPLACE Ssn        WITH m_ssn
         REPLACE Wkcenter   WITH m_wkcenter
         REPLACE Reg_wage   WITH m_reg_wage
      ENDIF

   CASE command = 'D'
      DELETE

   CASE command = 'U'
      RECALL

   ENDCASE
ENDDO
* eof empedit.prg
************************************************
```

```
* emplist.prg
* 7/18/87
PROCEDURE EMPLIST
STORE ' ' TO devout
STORE 1 TO pagenum
DO L23MENU WITH devout,'SPA',"Select device: "+;
"<P>rinter, <S>creen, <A>bort"
DO CASE
CASE devout = 'P'
   STORE 58 TO MAXLINES
   SET PRINT ON

CASE devout = 'S'
   STORE 23 TO MAXLINES
   SET PRINT OFF

CASE devout = 'A'
   RETURN

ENDCASE

SELECT A
SET INDEX TO EMPNAME, EMPLOYEE
?
?
?
?
? "                      LIST OF EMPLOYEES AS OF " +;
DTOC(DATE())+"        PAGE "+STR(pagenum)
?
? "               NAME                  SSN     "+;
" WAGE   OVER    GRADE   WC  CLOCKCODE"
?
STORE 7 TO LINECOUNT
GO TOP
DO WHILE .NOT. EOF()
   IF SUBSTR(GRADE,1,1) = 'G' .AND.;
   1.5*REG_WAGE > MAXGSOT
      STORE MAXGSOT TO OT_WAGE
   ELSE
      STORE INT(150*REG_WAGE + .5)/100.00 TO OT_WAGE
   ENDIF
   STORE STR(OT_WAGE,6,2) TO MOT_WAGE
```

```
   STORE STR(REG_WAGE,6,2) TO MREG_WAGE
   ? '     '+NAME,SSN,MREG_WAGE,MOT_WAGE+;
   '  ',GRADE,WKCENTER,CLOCKCODE
   LINECOUNT = LINECOUNT + 1
   IF LINECOUNT > MAXLINES
      pagenum = pagenum + 1
      IF DEVOUT = 'S'
         WAIT
         LINECOUNT = 1
      ELSE
         EJECT
         ?
         ?
         ?
         ? "                        LIST OF EMPLOYEES "+;
         "AS OF " + DTOC(DATE())+"       "+;
         "PAGE "+STR(pagenum)
         ?
         ? "                NAME                    "+;
         "SSN       WAGE    OVER     GRADE    WC "+;
         "CLOCKCODE"
         ?
         LINECOUNT = 8
      ENDIF
   ENDIF
   SKIP
ENDDO
IF DEVOUT = 'P'
   EJECT
ENDIF
SET INDEX TO EMPLOYEE, EMPNAME
SET PRINT OFF
RETURN
* EOF EMPLIST.PRG
************************************************
```

```
* FCREP
* 7/16/87
PROCEDURE FCREP

STORE ' ' TO device
STORE 1 TO pagenum
STORE LASTDATE - 13 TO FROMDATE

STORE LASTDATE TO TODATE

@ 20,0
@ 20,27 SAY "Enter first date to be included:"
@ 20,60 GET FROMDATE PICTURE '99/99/99'
@ 21,0
@ 21,27 SAY "Enter last date to be included:"
@ 21,59 GET TODATE PICTURE '99/99/99'
READ
STORE DTOC(FROMDATE) TO CFROMDAT
STORE DTOC(TODATE) TO CTODAT
STORE "NON-REIMBURSABLE CHARGES"+;
" &CFROMDAT TO &CTODAT" TO PERIOD

STORE 0.00 TO MTORGHR
STORE 0.00 TO MTORGCO
STORE 0.00 TO MTOOTHR
STORE 0.00 TO MTOOTCO

STORE ' ' TO devout
DO L23MENU WITH devout,'SPA',"Select device:"+;
"  <P>rinter, <S>creen, <A>bort"
DO CASE
CASE devout = 'P'
   STORE 58 TO MAXLINES
   SET PRINT ON

CASE devout = 'S'
   STORE 23 TO MAXLINES
   SET PRINT OFF

CASE devout = 'A'
   RETURN

ENDCASE

SELECT B
SET INDEX TO REPORT

?
?
?
?
```

```
? "                  "+PERIOD+"    "+;
"Page "+STR(pagenum,2)
?
? "   JOB NUMBER  REGULAR  REGULAR  OVERTIME  "+;
"OVERTIME   TOTAL   TOTAL"
? "                HOURS   CHARGES   HOURS    "+;
"CHARGES    HOURS  CHARGES"
?
STORE 10 TO LINECOUNT
GO TOP

DO WHILE .NOT. EOF()

   STORE LMC TO MLMC
   STORE 0.00 TO MFCRGHR
   STORE 0.00 TO MFCRGCO
   STORE 0.00 TO MFCOTHR
   STORE 0.00 TO MFCOTCO


   DO WHILE LMC=MLMC
      STORE JOBNO TO MJOBNO
      STORE 0.00 TO MJOBRGHR,MJOBRGCO,MJOBOTHR,;
      MJOBOTCO

      DO WHILE JOBNO=MJOBNO
         IF DATE >= FROMDATE .AND. ;
         DATE <= TODATE.AND. SEG = "    "

            MJOBRGHR = MJOBRGHR + REGHRS
            MJOBRGCO = MJOBRGCO +;
                       INT(100*REGHRS*REG_WAGE+.5)/100

            MJOBOTHR = MJOBOTHR + OTHRS
            MJOBOTCO = MJOBOTCO +;
                       INT(100*OT_WAGE*OTHRS+.5)/100
         ENDIF

         SKIP
      ENDDO

      IF MJOBRGCO+MJOBOTCO > 0.00
         ? "      "+MJOBNO, STR(MJOBRGHR,8,2),;
         STR(MJOBRGCO,9,2), STR(MJOBOTHR,7,2),;
         STR(MJOBOTCO,8,2),STR(MJOBRGHR+MJOBOTHR,9,2),;
         STR(MJOBRGCO+MJOBOTCO,10,2)

         linecount = linecount + 1
         IF LINECOUNT > MAXLINES
            pagenum = pagenum + 1
            IF DEVOUT = 'S'
```

```
            WAIT
            LINECOUNT = 1
         ELSE
            EJECT
            ?
            ?
            ?
            ?
            ? "                    "+PERIOD+;
            "   Page "+ STR(pagenum,2)
            ?
            ? "   JOB NUMBER  REGULAR  REGULAR  "+;
            "OVERTIME  OVERTIME   TOTAL   TOTAL"
            ? "                  HOURS   CHARGES   "+;
            "HOURS    CHARGES    HOURS  CHARGES"
            ?
            LINECOUNT = 10
         ENDIF
      ENDIF
   ENDIF

   MFCRGHR = MFCRGHR + MJOBRGHR
   MFCRGCO = MFCRGCO + MJOBRGCO
   MFCOTHR = MFCOTHR + MJOBOTHR
   MFCOTCO = MFCOTCO + MJOBOTCO

ENDDO

IF MFCRGCO+MFCOTCO > 0.00

   ? '  TOTAL '+MLMC, STR(MFCRGHR,8,2),;
   STR(MFCRGCO,9,2),STR(MFCOTHR,7,2),;
   STR(MFCOTCO,8,2),STR(MFCRGHR+MFCOTHR,9,2),;
   STR(MFCRGCO+MFCOTCO,10,2)
   ? "****************************************"+;
   "****************************************"
   ?
   LINECOUNT = LINECOUNT + 3
   IF LINECOUNT > MAXLINES
      pagenum = pagenum + 1
      IF DEVOUT = 'S'
         WAIT
         LINECOUNT = 1
      ELSE
         EJECT
         ?
         ?
         ?
         ?
         ? "                    "+PERIOD+"   Page "+;
```

```
            STR(pagenum,2)
            ?
            ? "    JOB NUMBER   REGULAR   REGULAR "+;

            "OVERTIME  OVERTIME   TOTAL   TOTAL"
            ? "                  HOURS   CHARGES   "+;
            "HOURS    CHARGES    HOURS  CHARGES"
            ?
            LINECOUNT = 10
         ENDIF
      ENDIF
   ENDIF

   MTORGHR = MTORGHR + MFCRGHR
   MTORGCO = MTORGCO + MFCRGCO
   MTOOTHR = MTOOTHR + MFCOTHR
   MTOOTCO = MTOOTCO + MFCOTCO

ENDDO {WHILE NOT EOF}
?
? "    **  GRAND TOTALS  **"
?
? "              "
?? STR(MTORGHR,8,2), STR(MTORGCO,9,2), STR(MTOOTHR,7,2),;
STR(MTOOTCO,8,2), STR(MTORGHR+MTOOTHR,9,2),;
STR(MTORGCO+MTOOTCO,10,2)

IF DEVOUT = 'S'
   WAIT
ELSE
   EJECT
ENDIF
DO REBYSEG
DO REBYLMC
RETURN

* EOP FCREP

***********************************************
```

```
PROCEDURE FILEMENU
* Date.....: 05/07/87
STORE ' ' TO devout
DO WHILE .T.

   * ---Display menu options, centered on the screen.
   *    draw menu border and print heading
   CLEAR
   @ 2, 0 TO 16,79 DOUBLE
   @ 3,25 SAY [F I L E   M A I N T E N A N C E]
   @ 4,1 TO 4,78 DOUBLE
   * ---display detail lines
   @  7,27 SAY [1. Edit Employee Records]
   @  8,27 SAY [2. Edit Timecard Data]
   @  9,27 SAY [3. Edit Job Order Data]
   @ 10,27 SAY [4. List Employee Data]
   @ 11,27 SAY [5. List Job Order Data]
   @ 12,27 SAY [6. Change Max GS OT and ]+;
   [Acceleration Rates]
   @ 14, 27 SAY '0. EXIT'
   STORE 0 TO selectnum
   @ 16,33 SAY " select     "
   @ 16,42 GET selectnum PICTURE "9" RANGE 0,6
   READ

   DO CASE
   CASE selectnum = 0
      RETURN

   CASE selectnum = 1
      *  DO Employee Record Data
      do empedit

   CASE selectnum = 2
      *  DO Timecard Data
      do timEDIT

   CASE selectnum = 3
      *  DO Fund Code Data
      do jobedit

   CASE selectnum = 4
      *  DO LIST EMPLOYEES
      DO EMPLIST

   CASE selectnum = 5
      *  DO LIST JOBS
      DO JOBLIST
```

```
   CASE selectnum = 6
      CLEAR
      hiaclmc = hiaclmc + SPACE(80)
      DO SAYGET WITH 10, "Enter maximum GS overtime"+;
      " rate:",maxgsot,'999.99'
      DO SAYGET WITH 11, "Enter high reimbursable "+;
      "acceleration percentage:",hiacrate,'99.9'
      DO SAYGET WITH 12, "Enter low reimbursable "+;
      "acceleration percentage:",loacrate,'99.9'
      DO MSG WITH 13, "Enter LMC's to which the high"+;

      " acceleration percentage applies:"
      DO SAYGET WITH 14, '',;
      hiaclmc,'!!!! !!!! !!!! !!!! !!!! !!!! !!!!'+;
      '!!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!!'
      READ
      hiaclmc = TRIM(hiaclmc)

   ENDCASE

ENDDO T
RETURN
* EOP FILEMENU
************************************************

* JOBDISP.PRG
PROCEDURE JOBDISP
IF DELETED()
   @ 4, 50 SAY "DELETED"
ELSE
   @ 4, 50 SAY "       "
ENDIF

@ 13, 29 SAY "JOB NUMBER ";
GET JOBNO PICTURE '9!!!!!'
@ 14, 29 SAY "LMC" GET LMC PICTURE '!!!!'
@ 15, 29 SAY "SEG#" GET SEG PICTURE '!!!!'

RETURN

* EOF JOBDISP.PRG
************************************************
```

```
* JOBDISPM.PRG
PROCEDURE JOBDISPM
IF DELETED()
   @ 4, 50 SAY "DELETED"
ELSE
   @ 4, 50 SAY "       "
ENDIF

@ 13, 29 SAY "JOB NUMBER ";
GET m_JOBNO PICTURE '9!!!!!'
@ 14, 29SAY "LMC" GET m_LMC PICTURE '!!!!'
@ 15, 29 SAY "SEG#" GET m_SEG PICTURE '!!!!'

RETURN

* EOF JOBDISPM.PRG
**********************************************
```

```
* JOBEDIT.PRG
PROCEDURE JOBEDIT
CLEAR
SELECT C
GO TOP
command = ' '
Opcomm = ' '

DO PAINTBOX WITH 6, 19, "J O B   F I L E"+;
"   M A I N T E N A N C E"

DO WHILE command <> 'E'
   DO JOBDISP
   CLEAR GETS
   DO L23MENU WITH command, 'FSBCADUE','<F>ind, '+;
   '<S>kip, <B>ack, <C>hange, <A>dd, <D>elete, '+;
   '<U>ndelete, <E>xit '

   DO CASE

   CASE command = 'F'
      key = '      '
      DO SAYGET WITH 23, 'Enter job number '+;
      'to find:',key,'9!!!!!'
      READ
      key = TRIM(key)
      SEEK key
      IF EOF()
         DO MSG WITH 22, '&key not found.'
         GO BOTTOM
      ENDIF

   CASE command = 'S'
      IF (EOF() .AND. BOF())
         DO MSG WITH 22, 'There are no records '+;
         'in the file.'
      ELSE
         IF EOF()
            GO TOP
         ELSE
            SKIP
         ENDIF
         IF EOF()
            GO TOP
         ENDIF
      ENDIF
```

```
   CASE command = 'B'
      IF (EOF() .AND. BOF())
         DO MSG WITH 22, 'There are no records '+;
         'in the file.'
      ELSE
         IF BOF()
            GO BOTTOM
         ENDIF
         SKIP -1
         IF BOF()
            GO BOTTOM
         ENDIF
      ENDIF

   CASE command = 'C'
      m_lmc = lmc
      m_seg = seg
      @ 14, 29 SAY "LMC" GET m_LMC PICTURE '!!!!'
      @ 15, 29 SAY "SEG#" GET m_SEG PICTURE '!!!!'
      READ
      DO L23MENU WITH Opcomm, 'SA', "<S>ave, <A>bort"
      IF Opcomm = 'S'
         REPLACE LMC WITH m_lmc, SEG with m_seg
      ENDIF
   CASE command = 'A'
      m_jobno = SPACE(  6)
      m_lmc = SPACE( 4)
      m_seg = SPACE( 4)
      DO JOBDISPM
      READ
      DO L23MENU WITH Opcomm, 'SA', "<S>ave, <A>bort"
      IF Opcomm = 'S'
         SEEK m_jobno
         IF FOUND()
            DO MSG WITH 22, 'This job number '+;
            'is already in the file.'
         ELSE
            APPEND BLANK
            REPLACE JOBNO WITH m_jobno,;
            LMC WITH m_lmc, SEG with m_seg
         ENDIF
      ENDIF
   CASE command = 'D'
      DELETE
   CASE command = 'U'
      RECALL
   ENDCASE
ENDDO
* eof jobedit.prg
**********************************************
```

```
* joblist.prg
* 7/18/87
PROCEDURE JOBLIST
STORE ' ' TO devout
DO L23MENU WITH devout,'SPA',"Select device:"+;
"  <P>rinter, <S>creen, <A>bort"
DO CASE
CASE devout = 'P'
   STORE 58 TO MAXLINES
   SET PRINT ON
CASE devout = 'S'
   STORE 23 TO MAXLINES
   SET PRINT OFF
CASE devout = 'A'
   RETURN
ENDCASE
SELECT C
?
?
?
?
? "                    LIST OF ACTIVE "+;
"JOBS AS OF " + DTOC(DATE())
?
? "       JOB NUMBER    LMC    SEG#"
?
STORE 7 TO LINECOUNT
GO TOP
DO WHILE .NOT. EOF()
   ? '          '+JOBNO+'    '+LMC+'     '+SEG
   LINECOUNT = LINECOUNT + 1
   IF LINECOUNT > MAXLINES
      IF DEVOUT = 'S'
         WAIT
         LINECOUNT = 1
      ELSE
         EJECT
         ?
         ?
         ?
         ?
         LINECOUNT = 5
      ENDIF
   ENDIF
   SKIP
ENDDO
SELECT B
SET PRINT OFF
RETURN
* EOF JOBLIST.PRG
**********************************************
```

```
* L23MENU
* 05/06/87
PROCEDURE L23MENU
PARAMETERS choice, possible, prompt
badans = .T.
@ 23,  1 SAY space(77)
DO WHILE badans
   @ 23, (78-LEN(prompt))/2 SAY prompt;
   GET choice PICTURE '!'
   READ
   IF choice $ possible
      STORE .F. TO badans
   ENDIF
ENDDO
* -- clear message line
@ 22,  1 SAY space(77)
RETURN
* EOP L23MENU
************************************************
```

```
* LOCDUP.PRG
* Duplication Check
* July 6, 1987
PROCEDURE LOCDUP

CLEAR
? "Be sure your printer is on."
WAIT
SET PRINT ON

? "Checking for duplicates in employee file..."
SELECT A
GO TOP
DO WHILE .NOT. EOF()
   Compare = SSN
   SKIP
   IF SSN = Compare
      * -- skip back one record
      SKIP -1
      * -- and list identical records
      LIST SSN, NAME WHILE SSN = Compare OFF
      ?
   ENDIF
ENDDO (while not eof)


? "Checking for duplicates in job order file..."
SELECT C
GO TOP
DO WHILE .NOT. EOF()
   Compare = JOBNO
   SKIP
   IF JOBNO = Compare
      * -- skip back one record
      SKIP -1
      * -- and list identical records
      ? " JOB NO  LMC  SEG#"
      LIST JOBNO, LMC, SEG WHILE JOBNO = Compare OFF
      ?
   ENDIF
ENDDO (while not eof)

SET PRINT OFF
RETURN
*EOP LOCDUP.PRG

**********************************************
```

```
* MAKEFILE
* 8/26/87
PROCEDURE MAKEFILE
SELECT B
USE TIMECARD INDEX TIMECARD
STORE LASTDATE TO MAKEDATE
STORE "OUTFILE1.TXT" TO OUTFILE
CLEAR
DO SAYGET WITH 12,"Enter date: ",;
MAKEDATE, "99/99/99"
READ
SEEK DTOC(MAKEDATE)
IF FOUND()
   DO SAYGET WITH 13,"Enter name of output file: ",;
   OUTFILE, "!!!!!!!!.!!!"
   READ
   IF .NOT. FILE(OUTFILE)
      COPY TO &OUTFILE FIELDS SSN,JOBNO,REGHRS,OTHRS;
      WHILE DATE = MAKEDATE SDF
      ? "File created successfully."
   ELSE
      ? "This file already exists.  Please "+;
        "name the file differently."
   ENDIF
ELSE
   ? "No records exist for the requested date."
ENDIF
CLOSE DATABASES
WAIT
RETURN
* EOP MAKEFILE


************************************************

* MSG
* 05/06/87
PROCEDURE MSG
PARAMETERS linenum, msgtxt
SET BELL ON
? CHR(7)
SET BELL OFF
@ linenum, 1 SAY space (77)
@ linenum, (80-LEN(msgtxt))/2 SAY msgtxt
RETURN

************************************************
```

```
* PAINTBOX
* 05/06/87
PROCEDURE PAINTBOX
PARAMETERS top, bottom, title
CLEAR
@ top, 0 TO bottom, 79 DOUBLE
@ top+1, (80-LEN(title))/2 SAY title
@ top+2, 1 TO top+2,78 DOUBLE
RETURN
* EOP PAINTBOX

***********************************************

* REBYLMC
* 8/13/87
PROCEDURE REBYLMC

STORE 1 TO pagenum
STORE "TOTAL ACCELERATED REIMBURSABLE CHARGES"+;
" BY LMC &CFROMDAT TO &CTODAT" TO PERIOD

STORE 0.00 TO MTORGHR
STORE 0.00 TO MTORGCO
STORE 0.00 TO MTOOTHR
STORE 0.00 TO MTOOTCO

SET INDEX TO REPORT

?
?
?
?
? PERIOD+"   Page "+STR(pagenum,2)
?
? "   LMC          REGULAR  REGULAR  OVERTIME "+;
" OVERTIME   TOTAL   TOTAL"
? "                HOURS   CHARGES   HOURS    "+;
"CHARGES    HOURS  CHARGES"
?
STORE 10 TO LINECOUNT
GO TOP

DO WHILE .NOT. EOF()

   STORE LMC TO MLMC
   STORE 0.00 TO MFCRGHR
   STORE 0.00 TO MFCRGCO
   STORE 0.00 TO MFCOTHR
   STORE 0.00 TO MFCOTCO
```

```
DO WHILE LMC=MLMC
   IF LMC $ HIACLMC
      STORE HIACRATE TO ACCRATE
   ELSE
      STORE LOACRATE TO ACCRATE
   ENDIF
   STORE JOBNO TO MJOBNO
   STORE 0.00 TO MJOBRGHR,MJOBRGCO,MJOBOTHR, MJOBOTCO

   DO WHILE JOBNO=MJOBNO
      IF DATE >= FROMDATE .AND.;
      DATE <= TODATE .AND. SEG <> "    "

         MJOBRGHR = MJOBRGHR + REGHRS
         MNEWRGCO = INT(100*REGHRS*REG_WAGE+.5)/100
         MACCELCO = INT(MNEWRGCO*ACCRATE+.5)/100
         MJOBRGCO = MJOBRGCO + MNEWRGCO + MACCELCO

         MJOBOTHR = MJOBOTHR + OTHRS
         MNEWOTCO = INT(100*OT_WAGE*OTHRS+.5)/100
         MOTREGCO = INT(100*REG_WAGE*OTHRS+.5)/100
         MACCELOT = INT(MOTREGCO*ACCRATE+.5)/100
         MJOBOTCO = MJOBOTCO + MNEWOTCO + MACCELOT
      ENDIF

      SKIP
   ENDDO

   MFCRGHR = MFCRGHR + MJOBRGHR
   MFCRGCO = MFCRGCO + MJOBRGCO
   MFCOTHR = MFCOTHR + MJOBOTHR
   MFCOTCO = MFCOTCO + MJOBOTCO

ENDDO

IF MFCRGCO+MFCOTCO > 0.00

   ? '  TOTAL '+MLMC, STR(MFCRGHR,8,2),;
   STR(MFCRGCO,9,2),STR(MFCOTHR,7,2),;
   STR(MFCOTCO,8,2),STR(MFCRGHR+MFCOTHR,9,2),;
   STR(MFCRGCO+MFCOTCO,10,2)
   ?
   LINECOUNT = LINECOUNT + 2
   IF LINECOUNT > MAXLINES
      pagenum = pagenum + 1
      IF DEVOUT = 'S'
         WAIT
         LINECOUNT = 1
      ELSE
         EJECT
         ?
```

```
            ?
            ?
            ?
            ? PERIOD+"   Page "+STR(pagenum,2)
            ?
            ? "     LMC       REGULAR  REGULAR  "+;
            "OVERTIME  OVERTIME   TOTAL   TOTAL"
            ? "                 HOURS   CHARGES  "+;
            " HOURS    CHARGES    HOURS  CHARGES"
            ?
            LINECOUNT = 10
         ENDIF
      ENDIF
   ENDIF

   MTORGHR = MTORGHR + MFCRGHR
   MTORGCO = MTORGCO + MFCRGCO
   MTOOTHR = MTOOTHR + MFCOTHR
   MTOOTCO = MTOOTCO + MFCOTCO

ENDDO (WHILE NOT EOF)
?
? "    **  GRAND TOTALS  **"
?
? "              "
?? STR(MTORGHR,8,2),STR(MTORGCO,9,2),STR(MTOOTHR,7,2),;
STR(MTOOTCO,8,2),STR(MTORGHR+MTOOTHR,9,2),;
STR(MTORGCO+MTOOTCO,10,2)

IF DEVOUT = 'S'
   WAIT
ELSE
   EJECT
   SET PRINT OFF
ENDIF
SET INDEX TO TIMECARD, REPORT, REPORT2
RETURN

************************************************
```

```
* REBYSEG
* 8/13/87
PROCEDURE REBYSEG

STORE 1 TO pagenum
STORE "ACCELERATED REIMBURSABLE CHARGES"+;
" BY SEG# &CFROMDAT TO &CTODAT" TO PERIOD

STORE 0.00 TO MTORGHR
STORE 0.00 TO MTORGCO
STORE 0.00 TO MTOOTHR
STORE 0.00 TO MTOOTCO

SET INDEX TO REPORT2

?
?
?
?
? "    "+PERIOD+"    Page "+STR(pagenum,2)
?
? "   JOB NUMBER  REGULAR  REGULAR  OVERTIME  "+;
"OVERTIME   TOTAL   TOTAL"
? "                HOURS   CHARGES   HOURS    "+;
"CHARGES    HOURS  CHARGES"
?
STORE 10 TO LINECOUNT
GO TOP

DO WHILE .NOT. EOF()

   STORE SEG TO MSEG
   STORE 0.00 TO MFCRGHR
   STORE 0.00 TO MFCRGCO
   STORE 0.00 TO MFCOTHR
   STORE 0.00 TO MFCOTCO


   DO WHILE SEG=MSEG
      STORE JOBNO TO MJOBNO
      STORE 0.00 TO MJOBRGHR,MJOBRGCO,MJOBOTHR,;
      MJOBOTCO

      DO WHILE JOBNO=MJOBNO
         IF DATE >= FROMDATE .AND.;
         DATE <= TODATE .AND. SEG <> "    "
            IF LMC $ HIACLMC
               STORE HIACRATE TO ACCRATE
            ELSE
               STORE LOACRATE TO ACCRATE
            ENDIF
```

```
            MJOBRGHR = MJOBRGHR + REGHRS
            MNEWRGCO = INT(100*REGHRS*REG_WAGE+.5)/100
            MACCELCO = INT(MNEWRGCO*ACCRATE+.5)/100
            MJOBRGCO = MJOBRGCO + MNEWRGCO + MACCELCO

            MJOBOTHR = MJOBOTHR + OTHRS
            MNEWOTCO = INT(100*OT_WAGE*OTHRS+.5)/100
            MOTREGCO = INT(100*REG_WAGE*OTHRS+.5)/100
            MACCELOT = INT(MOTREGCO*ACCRATE+.5)/100
            MJOBOTCO = MJOBOTCO + MNEWOTCO + MACCELOT
         ENDIF

         SKIP
      ENDDO

      IF MJOBRGCO+MJOBOTCO > 0.00
         ? "      "+MJOBNO, STR(MJOBRGHR,8,2),;
         STR(MJOBRGCO,9,2),STR(MJOBOTHR,7,2),;
         STR(MJOBOTCO,8,2),STR(MJOBRGHR+MJOBOTHR,9,2),;
         STR(MJOBRGCO+MJOBOTCO,10,2)

         linecount = linecount + 1
         IF LINECOUNT > MAXLINES
            pagenum = pagenum + 1
            IF DEVOUT = 'S'
               WAIT
               LINECOUNT = 1
            ELSE
               EJECT
               ?
               ?
               ?
               ?
               ? "   "+PERIOD+"   Page "+STR(pagenum,2)
               ?
               ? "   JOB NUMBER  REGULAR  REGULAR  "+;
               "OVERTIME  OVERTIME   TOTAL   TOTAL"
               ? "                  HOURS   CHARGES  "+;
               " HOURS    CHARGES    HOURS  CHARGES"
               ?
               LINECOUNT = 10
            ENDIF
         ENDIF
      ENDIF
      MFCRGHR = MFCRGHR + MJOBRGHR
      MFCRGCO = MFCRGCO + MJOBRGCO
      MFCOTHR = MFCOTHR + MJOBOTHR
      MFCOTCO = MFCOTCO + MJOBOTCO
   ENDDO
   IF MFCRGCO+MFCOTCO > 0.00
```

```
      ? '  TOTAL '+MSEG, STR(MFCRGHR,8,2),;
      STR(MFCRGCO,9,2),STR(MFCOTHR,7,2),;
      STR(MFCOTCO,8,2),STR(MFCRGHR+MFCOTHR,9,2),;
      STR(MFCRGCO+MFCOTCO,10,2)
      ? "**************************************"+;
      "**************************************"
      ?
      LINECOUNT = LINECOUNT + 3
      IF LINECOUNT > MAXLINES
         pagenum = pagenum + 1
         IF DEVOUT = 'S'
            WAIT
            LINECOUNT = 1
         ELSE
            EJECT
            ?
            ?
            ?
            ?
            ? "   "+PERIOD+"   Page "+STR(pagenum,2)
            ?
            ? "   JOB NUMBER  REGULAR  REGULAR  "+;
            "OVERTIME  OVERTIME   TOTAL   TOTAL"
            ? "                  HOURS   CHARGES  "+;
            " HOURS    CHARGES    HOURS  CHARGES"
            ?
            LINECOUNT = 10
         ENDIF
      ENDIF
   ENDIF
   MTORGHR = MTORGHR + MFCRGHR
   MTORGCO = MTORGCO + MFCRGCO
   MTOOTHR = MTOOTHR + MFCOTHR
   MTOOTCO = MTOOTCO + MFCOTCO
ENDDO {WHILE NOT EOF}
?
? "    **  GRAND TOTALS  **"
?
? "              "
?? STR(MTORGHR,8,2),STR(MTORGCO,9,2),STR(MTOOTHR,7,2),;
STR(MTOOTCO,8,2), STR(MTORGHR+MTOOTHR,9,2),;
STR(MTORGCO+MTOOTCO,10,2)
IF DEVOUT = 'S'
   WAIT
ELSE
   EJECT
ENDIF
SET INDEX TO TIMECARD, REPORT, REPORT2
RETURN
* EOP REBYSEG
**********************************************
```

```
* SAYGET
* 05/06/87
PROCEDURE SAYGET
PARAMETERS lineno,saythis,getthis,template
@ lineno, 1 SAY space(77)
@ lineno, (79-LEN(saythis)-LEN(template))/2 SAY ;
saythis GET getthis PICTURE template
RETURN
* EOP SAYGET

***********************************************

PROCEDURE TAENTEMP
STORE space(9) to M_ssn
DO SAYGET WITH 11, 'SSN', M_ssn, '999999999'
READ
SELECT A
FIND "&M_ssn"
* -- macro substitution in quotes to accept blank
* -- without bombing
IF EOF()
   DO MSG WITH 22, "SSN not listed in employee file."
   STORE .F. TO ssn_ok
ELSE
   STORE reg_wage TO m_reg_wage
   IF SUBSTR(A->GRADE,1,1) = 'G' .AND.;
   INT(m_reg_wage * 150 + .5)/100.00 > Maxgsot
      STORE Maxgsot to m_otwage
   ELSE
      STORE INT(m_reg_wage * 150 + .5)/100.00 ;
      to m_ot_wage
   ENDIF
   STORE .T. TO ssn_ok
   DO TAENTJOB
ENDIF
SELECT B
RETURN
* eof taentemp
***********************************************
```

```
PROCEDURE TAENTJOB
STORE " " TO command2
STORE " " TO command3
STORE "      " to M_jobno
STORE 0 TO M_reghrs
STORE 0 TO M_othrs
STORE .F. TO jobdone
DO WHILE .NOT. jobdone
   DO SAYGET WITH 12, "Job Order Number",;
   M_jobno,"!!!!!!"
   READ
   SELECT C
   FIND "&M_jobno"
   IF EOF()   && i.e. number not in job file
      SELECT B
      DO MSG with 22, "This job order is"+;
      " not in the jobs file."
      DO L23MENU WITH command2, "AR",;
      "<R>e-enter job number, <A>bandon entry "
      IF command2 = 'A'
         STORE .T. TO jobdone
      ENDIF
   ELSE  && JO in file so get lmc and seg
      STORE LMC TO M_lmc
      STORE SEG TO M_seg
      SELECT B
      DO SAYGET WITH 13, "Enter regular hours worked",;
      M_reghrs, "999.99"
      DO SAYGET WITH 14, "Enter overtime "+;
      "hours worked",M_othrs, "999.99"
      READ
      DO L23MENU WITH command3, "SCA",;
      "<S>ave, <C>hange, <A>bandon"
      DO CASE
      CASE command3 = 'S'
         APPEND BLANK
         REPLACE reghrs WITH M_reghrs,;
         othrs WITH M_othrs, date with M_date, jobno;
         with M_jobno, reg_wage with M_reg_wage, lmc;
         with M_lmc,ssn with M_ssn, ot_wage with;
         M_ot_wage, seg with M_seg
         STORE .T. to jobdone
      CASE command3 = 'A'
         * -- exit loop
         STORE .T. to jobdone
      ENDCASE
   ENDIF (JO not in file)
ENDDO
RETURN
* EOF TAENTJOB
**********************************************
```

```
PROCEDURE TIMDISP
IF DELETED()
   @ 4, 50 SAY "DELETED"
ELSE
   @ 4, 50 SAY "       "
ENDIF

@ 10, 24 SAY "DATE ";
GET DATE PICTURE '99/99/99'
@ 11, 24 SAY "SOCIAL SECURITY NUMBER ";
GET SSN PICTURE '999999999'
@ 12, 24 SAY "JOB NUMBER ";
GET JOBNO PICTURE '9!!!!!'
@ 13, 24 SAY "REGULAR HOURS ";
GET REGHRS PICTURE '999.99'
@ 14, 24 SAY "OVERTIME HOURS ";
GET OTHRS PICTURE '999.99'
@ 15, 24 SAY "REGULAR WAGE ";
GET REG_WAGE PICTURE '999.99'
@ 16, 24 SAY "OVERTIME WAGE ";
GET OT_WAGE PICTURE '999.99'
@ 17, 24 SAY "LMC ";
GET LMC PICTURE '!!!!'
@ 18, 24 SAY "SEG# ";
GET SEG PICTURE '!!!!'

RETURN

* EOF TIMDISP.PRG
************************************************
```

```
PROCEDURE TIMDISPM
IF DELETED()
   @ 4, 50 SAY "DELETED"
ELSE
   @ 4, 50 SAY "       "
ENDIF

@ 10, 24 SAY "DATE ";
GET m_DATE PICTURE '99/99/99'
@ 11, 24 SAY "SOCIAL SECURITY NUMBER ";
GET m_SSN PICTURE '999999999'
@ 12, 24 SAY "JOB NUMBER ";
GET m_JOBNO PICTURE '9!!!!!'
@ 13, 24 SAY "REGULAR HOURS ";
GET m_REGHRS PICTURE '999.99'
@ 14, 24 SAY "OVERTIME HOURS ";
GET m_OTHRS PICTURE '999.99'
@ 15, 24 SAY "REGULAR WAGE ";
GET m_REG_WAGE PICTURE '999.99'
@ 16, 24 SAY "OVERTIME WAGE ";
GET m_OT_WAGE PICTURE '999.99'
@ 17, 24 SAY "LMC ";
GET m_LMC PICTURE '!!!!!'
@ 18, 24 SAY "SEG# ";
GET m_SEG PICTURE '!!!!!'

RETURN

* EOF TIMDISPM.PRG
*********************************************

PROCEDURE TIMEADD2

* -- initialize everything
STORE LASTDATE to M_date
STORE "         " to M_ssn
STORE "      " to M_jobno
STORE 0 to M_reghrs
STORE 0 to M_othrs
STORE 0 to M_reg_wage
STORE 0 to M_ot_wage
STORE 'E' to command
STORE .F. to ssn_ok
STORE .F. to job_ok

DO PAINTBOX WITH 5, 17, "A D D   L A B O R"+;
"   C A R D   D A T A"

@ 10,33 SAY "Date";
GET M_date PICTURE "99/99/99"
@ 11,33 SAY 'SSN';
```

```
GET M_ssn PICTURE '999999999'
@ 12,28 SAY "Job Order Number";
GET M_jobno PICTURE "!!!!!!"
@ 13,23 SAY "Enter regular hours worked";
GET M_reghrs PICTURE "999.99"
@ 14,23 SAY "Enter overtime hours worked";
GET M_othrs PICTURE "999.99"
CLEAR GETS


SET CONFIRM OFF
SELECT B

DO WHILE command <> 'Q'
   DO L23MENU WITH command, 'DSJQ',;
   "<D>ate, <S>sn, <J>ob, <Q>uit"
   DO CASE

   CASE command = 'D'
      DO SAYGET WITH 10, "Date", M_date, "99/99/99"

   CASE command = 'S'
      * -- enter all timecard data
      DO TAENTEMP

   CASE command = 'J' .AND. ssn_ok
      * -- enter job order and hours worked
      DO TAENTJOB

   CASE command = 'J' .AND. .NOT. ssn_ok
      DO MSG WITH 22,"You must have a "+;
      "validated SSN before invoking this."

   ENDCASE
ENDDO {while command <> 'Q'}
STORE M_date TO LASTDATE
RETURN
* EOF TIMEADD2
**********************************************
```

```
PROCEDURE TIMEDIT
CLEAR
SELECT B
GO TOP
command = ' '
Opcomm = ' '

DO PAINTBOX WITH 5, 21, "T I M E   C A R D"+;
"   F I L E   M A I N T E N A N C E"

DO WHILE command <> 'E'
   DO TIMDISP
   CLEAR GETS
   DO L23MENU WITH command, 'FSBCDUE','<F>ind, '+;
   '<S>kip, <B>ack, <C>hange, <D>elete, <U>ndelete,'+;
   ' <E>xit'

   DO CASE

   CASE command = 'F'
      STORE DTOC(DATE()) TO keydate
      DO SAYGET WITH 23, 'Enter date of time'+;
      ' card to find:',keydate, '99/99/99'
      READ

      key = '          '
      DO SAYGET WITH 23, 'Enter employee SSN:',;
      key, '999999999'
      READ
      key = TRIM(key)
      key = keydate + key
      SEEK key
      IF EOF()
         DO MSG WITH 22, 'Timecard not found.'
         GO BOTTOM
      ENDIF

   CASE command = 'S'
      IF (EOF() .AND. BOF())
         DO MSG WITH 22, 'There are no records '+;
         'in the file.'
      ELSE
         IF EOF()
            GO TOP
         ELSE
            SKIP
         ENDIF
         IF EOF()
            GO TOP
         ENDIF
      ENDIF
```

```
   CASE command = 'B'
      IF (EOF() .AND. BOF())
         DO MSG WITH 22, 'There are no records '+;
         'in the file.'
      ELSE
         IF BOF()
            GO BOTTOM
         ENDIF
         SKIP -1
         IF BOF()
            GO BOTTOM
         ENDIF
      ENDIF

   CASE command = 'C'
      m_jobno    = Jobno
      m_lmc      = Lmc
      m_seg      = Seg
      m_ssn      = Ssn
      m_date     = Date
      m_ot_wage  = Ot_wage
      m_othrs    = Othrs
      m_reg_wage = Reg_wage
      m_reghrs   = Reghrs
      DO TIMDISPM
      READ
      DO L23MENU WITH Opcomm, 'SA', "<S>ave, <A>bort"
      IF Opcomm = 'S'
         REPLACE Jobno      WITH m_jobno
         REPLACE Lmc        WITH m_lmc
         REPLACE Seg        WITH m_seg
         REPLACE Ssn        WITH m_ssn
         REPLACE Date       WITH m_date
         REPLACE Ot_wage    WITH m_ot_wage
         REPLACE Othrs      WITH m_othrs
         REPLACE Reg_wage   WITH m_reg_wage
         REPLACE Reghrs     WITH m_reghrs
      ENDIF
   CASE command = 'D'
      DELETE

   CASE command = 'U'
      RECALL
   ENDCASE
ENDDO
RETURN
* EOP TIMEDIT

* EOF TIMEPROC.PRG
```

LIST OF REFERENCES

1. Naval Postgraduate School Course Catalog, Naval Postgraduate School, Monterey, California, Academic Year 1987.

2. Budget Guidance Manual, NAVCOMPT Instruction 7102.2, 27 April 1983.

BIBLIOGRAPHY

Bohl, Marilyn, Information Processing, Science Research Associates, Chicago, 1984.

Burns, R. N. and Dennis, A. R., "Selecting the Appropriate Application Development Methodology", Data Base, Vol. 17, No. 1, Fall 1985, pp. 19-23.

Carrabis, Joseph-David, dBASE III PLUS: The Complete Reference, Osborne McGraw-Hill, Berkeley, 1987.

Castro, Luis A., Hanson, Jay, and Rettig, Tom, Advanced Programmer's Guide, Ashton-Tate, Culver City, California, 1985.

Davis, William S., Systems Analysis and Design, Addison-Wesley, Reading, Massachusetts, 1983.

Good, Michael D., Whiteside, John A., Wixon, Dennis R., and Jones, Sandra J., "Building a User-Derived Interface", Communications of the Association for Computing Machinery, Vol. 27, No. 10, October 1984, pp. 1032-1043.

International Business Machines Corporation, "Business Systems Planning Information Systems Planning Guide", International Business Machines Corporation, Armonk, New York, 1984.

Kroenke, David M., Database Processing, Science Research Associates, Chicago, 1983.

Martin, James, Application Development Without Programmers, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1982.

Page-Jones, Meilir, The Practical Guide to Structured Systems Design, Yourdon Press, Englewood Cliffs, New Jersey, 1980.

Yourdon, Edward, Managing the Structured Techniques, Yourdon Press, New York, 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chief of Naval Operations Director, Information Systems (OP-945) Navy Department Washington, D.C. 20350-2000	1
4.	Naval Facilities Engineering Command Program Manager, Public Works Centers (15) 200 Stovall Street Alexandria, Virginia 22332-2300	1
5.	Naval Facilities Engineering Command Director, Facilities Division (Code 100) 200 Stovall Street Alexandria, Virginia 22332-2300	1
6.	Naval Construction Battalion Center Director, Civil Engineer Support Office (Code 15) Port Hueneme, CA 93043	1
7.	Norman R. Lyons Administrative Sciences Department (Code 54LB) Naval Postgraduate School Monterey, California 93943-5000	1
8.	Taracad R. Sivasankaran Administrative Sciences Department (Code 54SJ) Naval Postgraduate School Monterey, California 93943-5000	1
9.	Public Works Officer, Code 43 Naval Postgraduate School Monterey, California 93943	1

10. Comptroller, Code 002 1
Naval Postgraduate School
Monterey, California 93943

11. David P. Dinwiddie, LT, CEC, USNR 2
Naval Construction Battalion Center
Director, Civil Engineer Support Office (Code 15)
Port Hueneme, CA 93043

Thesis
D57649 Dinwiddie
c.1 A database system for monitoring labor costs in a Public Works environment.